M 2022

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DEVELOPMENT OF NIR MODELS TO FULL UREA-FORMALDEHYDE RESIN CHARACTERIZATION

IGOR RAFAEL BRANDÃO CARVALHO

MASTER THESIS IN CHEMICAL ENGINEERING
PRESENTED TO THE FACULTY OF ENGINEERING
OF THE UNIVERSITY OF PORTO

Master in Chemical Engineering

Development of NIR models to full urea-formaldehyde resin characterization

Master dissertation

of

Igor Rafael Brandão Carvalho

Developed within the course of dissertation

held in

Sonae Arauco

Taking wood further

Supervisor at FEUP: **Prof. Fernando Gomes Martins**

Coordinator at Sonae Arauco: **Dr. Nádia Paiva**

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

DEPARTAMENTO DE
ENGENHARIA QUÍMICA

July 2022

Acknowledgment

I want to thank and dedicate this work to everyone who directly or indirectly contributed to the outcome of this master's thesis.

I firstly start by thanking my academic supervisor, professor Fernando Martins, for the invaluable patience, feedback, and trust throughout the dissertation development. I also thank Sonae Arauco for the opportunity to contribute to the company, and my enterprise supervisor, Dr. Nádia Paiva, for the knowledge and expertise transmitted. Also, I thank Dr. João Ferra for the support and constructive suggestions.

During these months, I had the opportunity to meet some friends that worked with me, which I must mention and especially thank. I want to thank Roberto Magalhães and Beatriz Nogueira for the technical assistance during the data and spectra acquisition and Mariana Santos and Sofia Gonçalves for their friendship and support.

Lastly, I dedicate this and all my achievements to my beloved mother, Selene Brandão, and my dear brother, Francys Campelo. Without them, nothing would be possible.

Igor Carvalho

Professor Fernando Martins, supervisor of this dissertation is an integrated member of LEPABE – Laboratory for Process Engineering, Environment, Biotechnology and Energy, financially supported by LA/P/0045/2020 (ALiCE), UIDB/00511/2020 and UIDP/00511/2020 (LEPABE), funded by national funds through FCT/MCTES (PIDDAC).

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

REPÚBLICA PORTUGUESA

Abstract

Nowadays, it is constant the search for methods capable of improving the efficiency of industrial processes. The combination of near-infrared spectroscopy (NIRS) and chemometrics emerges as one of the main tools to replace expensive and time-consuming techniques.

This dissertation presents NIRS-based models for predicting the principal amino resin properties, such as formaldehyde-urea molar ratio (FUMR), melamine content, and solids content (SC). The models will be applied in a Sonae Arauco factory in Germany, a wood-based panel production plant, to guarantee the quality of the received resins, replace time-consuming experimental techniques, reduce costs, and avoid problems in the panels' production.

The models were based on three regression types, partial least squares (PLS), artificial neural network (ANN), and support vector machines regression (SVR), and developed in two different platforms: an open-source using programming language and libraries in Python and a commercial software (OPUS version 6.5) used by the company to acquire spectra, create and apply PLS models.

Each regression method was studied differently based on the most representative spectral zones. The PLS-based models were created in OPUS and Python using different preprocessing methods to eliminate irrelevant information. Due to Python's versatility, several preprocessing combinations were tested for the PLS models, along with the creation of ANN and SVR models. The best models were selected according to their cross-validation performance through the coefficient of determination (R^2), and root mean squared error (RMSE). Additionally, after selection, the best ones were tested regarding an independent dataset as a final validation.

Python models had the best performance, especially those created with SVR for predicting FUMR and SC. R^2 values of cross-validation and test validation were above 0.95 with minimum RMSE. The PLS model for melamine content, also created in Python, performed well with good prediction errors but was not comparable to the previous ones due to the absence of representative bands related to this property. With R^2 of cross-validation and test validation values of 0.79 and 0.85, respectively, the melamine content model had a low RMSE of around 0.1. Despite the models created in Python having the best performance, the difference regarding those made in OPUS was not much significant.

Keywords: amino resins; chemometrics; near-infrared spectroscopy; regression models.

Resumo

Atualmente, é constante a procura por métodos capazes de melhorar a eficiência dos processos industriais. A combinação da espectroscopia do infravermelho próximo (NIRS) junto com a quimiometria surge como uma das principais ferramentas para a substituição de técnicas dispendiosas e demoradas.

Esta dissertação tem como objetivo apresentar modelos baseados em NIRS para a previsão das principais propriedades de resinas amino, como razão molar de formaldeído-ureia (FUMR), conteúdo de melamina e conteúdo de sólidos (SC). Esses modelos serão aplicados numa fábrica de produção de painéis de derivados de madeira da Sonae Arauco na Alemanha, podendo ser utilizados para garantir a qualidade das resinas recepcionadas, substituir técnicas experimentais demoradas, reduzir custos e evitar problemas na produção de painéis de madeira.

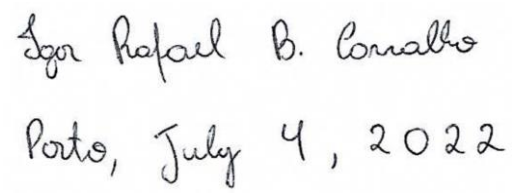
Os modelos criados foram baseados em três tipos de regressão, regressão por mínimos quadrados parciais (PLS), rede neuronal artificial (ANN) e regressão por máquinas de vetores de suporte (SVR), e desenvolvidos em duas plataformas diferentes: uma de código aberto, fazendo o uso da linguagem de programação e bibliotecas em Python e a outra um software comercial (OPUS versão 6.5) utilizado na aquisição de espectros, criação e aplicação de modelos PLS.

Utilizando as zonas espectrais mais representativas, cada metodologia de regressão foi estudada de forma distinta. Os modelos baseados em PLS foram criados em OPUS e em Python, utilizando diferentes métodos de pré-processamento com o intuito de eliminar informações irrelevantes. Devido a maior versatilidade do Python, foram testadas diversas combinações de pré-processamentos para os modelos em PLS, além da criação dos modelos ANN e SVR. Os melhores modelos foram selecionados de acordo com o seu desempenho em relação a validação cruzada através do coeficiente de determinação (R^2) e da raiz quadrada do erro quadrático médio (RMSE). Além disso, depois de selecionados, foram testados com o intuito de avaliar seu aproveitamento em relação a um conjunto de dados independentes.

Os modelos desenvolvidos em Python tiveram melhores desempenhos, principalmente aqueles criados com SVR para a previsão da FUMR e para o SC. Os valores dos R^2 para a validação cruzada e validação teste foram acima de 0.95 com RMSE mínimos. O modelo PLS para o conteúdo de melamina, também em Python, teve um bom desempenho com bons erros de previsão, mas não comparável aos anteriores devido à ausência de bandas representativas ou mais restritivas. Apesar dos valores de R^2 para a validação cruzada e validação teste serem de 0.79 e 0.85, respetivamente, os valores de RMSE obtidos para o conteúdo de melamina foram considerados baixos em torno de 0.1. Apesar de haver o destaque em relação aos modelos criados em Python, a diferença em relação aos modelos criados em OPUS não foi tão significativa.

Declaration

I hereby declare, under word of honour, that this work is original and that all non-original contributions is indicated and due reference is given to the author and source



Index

List of Figures

List of Tables

Notation and Glossary

List of Abbreviations

ANN	Artificial neural network
Ar	Aromatic ring
C	Regularization parameter
COE	Constant offset elimination
CV	Cross-validation
F	Formaldehyde
fiPLS	Forward interval partial least squares
FTIR	Fourier-transform infrared spectroscopy
FUMR	Formaldehyde urea molar ratio
GC	Gas chromatograph
HPLC	High performance liquid chromatography
iPLS	Interval partial least squares
IQR	Interquartile range
LC-MS	Liquid chromatography–mass spectrometry
looVal	Leaving-one-out cross-validation
LV	Latent variable
MDF	Medium density fiberboard
MF	Melamine-formaldehyde
MLR	Multiple linear regression
MS	Mass spectrometry
MSC	Multiplicative scatter correction
MUF	Melamine-urea-formaldehyde
NIPPY	Semi-automated preprocessing model for NIRS data
NIR	Near-infrared
NIRS	Near-infrared spectroscopy
NMR	Nuclear magnetic resonance
NW	Norris-Williams derivation
OSB	Oriented strand boards
PB	Particleboards
PC	Principal component
PCA	Principal component analysis
PCR	Principal Component regression
PF	Phenol-formaldehyde
PLS	Partial least squares
R^2	Coefficient of determination
R^2Cal	Coefficient of determination of calibration
R^2CV	Coefficient of determination of cross-validation
R^2Test	Coefficient of determination of test validation
RBF	Radial basis function
RMSE	Root mean squared error
RMSECal	Root mean squared error of calibration

RMSECV	Root mean squared error of cross-validation
RMSETest	Root mean squared error of test validation
RNV	Robust normal variate
SC	Solids content
SG	Savitzky-Golay derivation
siPLS	Synergy interval partial least squares
SLS	Straight-line subtraction
SNV	Standard Normal Variate
SVM	Support vector machines
SVR	Support vector machines regression
U	Urea
UF	Urea-formaldehyde
UV-VIS	Ultraviolet-visible spectroscopy
XES	X-ray emission spectroscopy

List of Symbols

A	Absorbances matrix
B	Coefficient matrix in partial least squares regression
E	Residuals related to x variables in partial least squares regression
E_A	Residual spectra matrix
E_Y	Error matrix in principal components regression
F	Residual error in partial least squares regression model
I	Information transmitted by a neuron in artificial neural network
N	The number of calibration or test spectra
P	Loadings in partial least squares regression
S	Scores matrix
T	Transposed of matrix
U	Scores related to y variables in partial least squares regression
V	Eigenvector matrix
W	Weight attributed to each neuron in artificial neural network
X	Input matrix
Y	Output/reference values matrix
$\bar{x}$	Average spectrum
β	Weights (or gains) in support vector machines regression
ε	Margin (or decision boundaries) in support vector machines regression
θ	Offset (or bias term) of the neurons in artificial neural network
ξ	Distance of the slack variables from the decision boundaries in support vector machines regression
a_0	Average value of the sample spectrum
a_1	Standard deviation of a sample spectrum
b_0	Interception parameter of a linear fit
b_{ref}	Slope of polynomial fit
c_0	Interception parameter of a linear fit in support vector machines regression
e	Error or residual error related to y variables in partial least squares regression
m	Smoothing points window
q	Loading vector related to y variables in partial least squares regression

s	Standard deviation of the data set
x	Input of a function
x_{corr}	Spectrum corrected by a preprocessing
x_{org}	Original spectrum without preprocessing
x_{ref}	Reference spectrum
x_{smooth}	Spectrum smoothed by a preprocessing
y	Output of a function
y_i	Reference value
$\hat{y}_i$	Predicted value
$\overline{y_i}$	Average of the reference values.

1 Introduction

1.1 Framing and presentation of the work

Nowadays, the increasing demand for quality improvement, product rationalization, sustainability, and more restricted expenditure control, potentiates the search for effective analysis methods with high performance and capable of guaranteeing the quality of the final product. Cheaper, faster, and easily operated analysis methods have appeared like near-infrared spectroscopy (NIRS) to assure industrial quality control and process monitoring.

NIRS and chemometrics have become crucial tools in developing prediction models in different industrial fields. In the wood panels industry, like in Sonae Arauco at the Portugal resins production plant, the NIR method has been used to predict important properties of the resins used in the agglutination of the wood particles.

The objective of this dissertation is to develop new models based on NIRS to predict resin properties, like formaldehyde urea molar ratio (FUMR), melamine and solids contents, to be applied in the Sonae Arauco plant in Germany, where external producers make the resin supply. The methods will be used to ensure the resin's quality, avoid storage contamination and production failures.

1.2 Presentation of the company

In 2016, Sonae Indústria started a strategic partnership with Arauco, resulting in a Joint Venture between two leading world companies in the wood sector. The combination of both companies gives rise to Sonae Arauco, one of the world's top producers of wood-based panels. The company's portfolio ranges from simple panels, like particleboards (PBs), to the more polished ones, like medium-density fiberboards (MDFs) and Oriented Strand Boards (OSBs) [1].

Along with the production of wood panels, Sonae Arauco's activity also includes the manufacture of formaldehyde-based resins by EuroResinas – Indústrias Químicas S.A., a company within the joint venture where four main resin types are manufactured: Phenol-Formaldehyde (PF), Urea-Formaldehyde (UF), Melamine-Formaldehyde (MF), and Melamine-Urea-Formaldehyde (MUF). Also, Euroresinas' activity includes producing decorative paper impregnated with MF, and Kraft paper impregnated with PF.

Sonae Arauco factories are presented in four countries: Portugal, Spain, South Africa, and Germany. The resin supply in Portugal and Spain is made by EuroResinas, while external manufacturers supply South Africa and Germany factories. Thus, Sonae Arauco guarantees the production quality in the Iberian countries since it is also responsible for producing the resins. While in the other countries, it depends on a third party to assure the quality of the resins and

avoid failures in the panels' production. So, it is crucial to analyze the resin before its reception to ensure it has the intended specifications.

1.3 Contribution of the author to the work

During the internship, I created different regression models using a commercial software (OPUS) and Python programming. The models developed in both platforms were robust with good predictive ability providing good results in terms of predictive statistical parameters (RMSE and R^2). The company will use the OPUS models in Germany to maintain the quality of the received resins. They can substitute standard time-consuming techniques and guarantee that the resins' FUMR, melamine content, and solids content obey the correct specifications.

1.4 Organization of the dissertation

The dissertation is composed of 7 chapters. The objective of this work, along with the presentation of the company, is shown in Chapter 1. In Chapter 2, the context and state of the art are presented to address the important information necessary to develop this dissertation. This chapter presents the main resin types and properties, followed by the methods of analysis, data preprocessing, and the most common regression methods. In Chapter 3, materials and methods, the procedure for creating the models is presented. The methodology to analyze the resin and acquire the spectra is shown, along with the validation and evaluation approach. In Chapter 4, the prominent spectrum bands discussion and the models with the best results are presented. Chapters 5 and 6 present the work's conclusion and assessment, followed by the references in Chapter 7.

2 Context and State of the art

2.1 Formaldehyde resins

Formaldehyde resins are the most used adhesives in the agglutination of wood particles in the development of panels like particleboard, plywood, and medium-density fiberboard. The most common resins differ in terms of their use and production. Adhesives based on melamine (MF and MUF), with more than 5 % (w/w) of melamine content, are used in the manufacture of external or semi-external wood panels due to their water and weather resistance. In comparison, UF resins are used for internal purposes due to their low moisture resistance and can be fortified by the addition of a low amount of melamine [2]. Due to the presence of -NH_2 groups in urea and melamine, these resins are classified as amino resins. The strong advantage regarding them is the fact they are thermosetting. In other words, they can change irreversibly, by a curing process, into an infusible and insoluble polymer network after applying heat or suitable radiation [3].

The production of UF resin is based on two steps, as shown in Figure 1. First, the alkaline condensation promotes the formation of mono-, di-, trimethyloureas by the nucleophilic attack in basic media. Then, the second step is the production of crosslinked resins with methylene-ether bridges by acid condensation (dehydration reaction) of methyloureas [2].

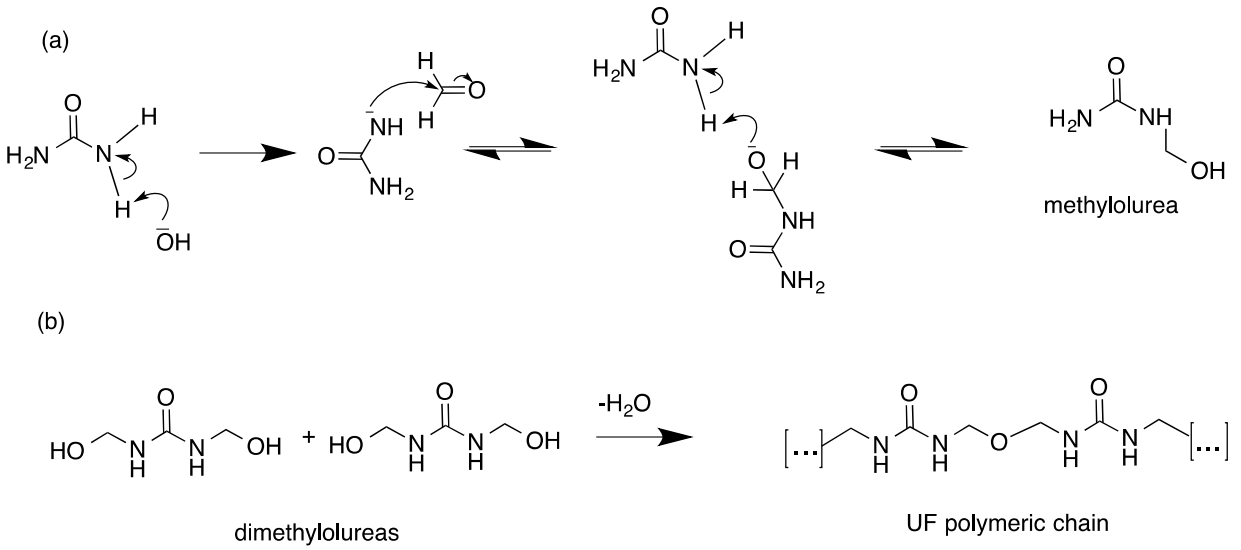


Figure 1 Alkaline condensation of urea and formaldehyde (a) and acid condensation of dimethylolureas (b) (adapted from [2]).

The manufacturing process of MUF and melamine-fortified UF resins are similar to UF resins and important due to the low formaldehyde emission [4]. The melamine allows better polymer stabilization and improves physical properties in wood panels [2]. In addition to the formation of methylolurea, the MUF production process also promotes the formation of

methylolmelamines and consequently creates a more ramified polymeric chain, as shown in Figure 2.

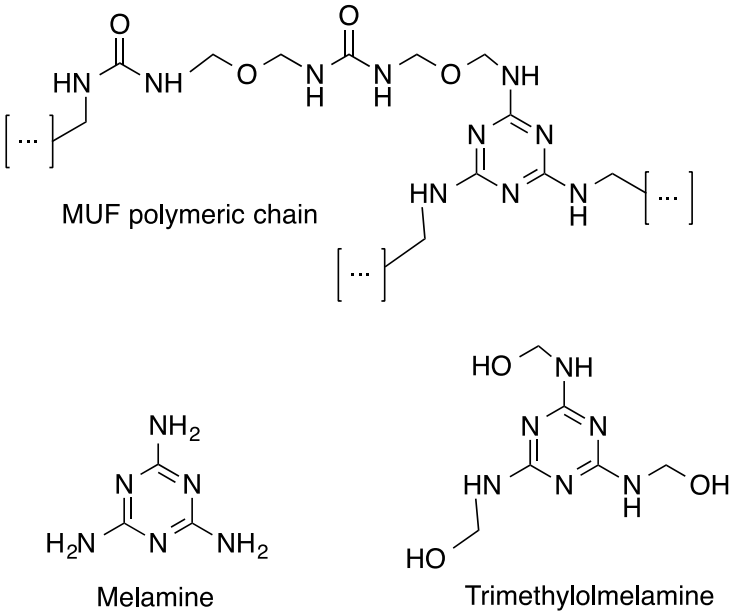


Figure 2 Main components presented in MUF resins

The most common UF production method is based on adding urea to the reaction medium. Initially, it consists of the reaction between excess formaldehyde with urea, in a proportion around 2.0:1 - 2.2:1 and under reflux. This first step can be done with a temperature of around 90 – 95 ˚C to decrease the amount of time, speeding up the reaction. Due to the exothermic characteristics, the alkaline condensation is completed after 10 - 30 min when the temperature attenuates. After, acid is added to the reaction media to decrease pH (5.0 - 5.3) and start the UF polymer building stage. When the desired viscosity is reached, pH is increased to stop the polymeric reaction, and the resin is cooled at 25 – 30 ˚C. The next step is the addition of urea to consume the excess formaldehyde at room temperature for 24 hours until the U/F molar ratio is in the correct range (1:1.1 - 1:1.7). After, the resin solution can be dried in a vacuum distiller to eliminate the excess water until the solids content is around 64 – 65 % [2].

2.2 Important formaldehyde resin properties

2.2.1 F/U molar ratio

Due to the concerns regarding the effects of formaldehyde on human health, resin producers were invited to decrease the emission of this organic compound. Therefore, the performance and properties of the resins were changed mainly the F/U molar ratio, which is related to the amount of free formaldehyde and crosslinking degree. Fortunately, due to research, current resins have the same or more performance as many years ago, with less formaldehyde emission and a lower F/U molar ratio [5].

2.2.2 Melamine content

As mentioned before, melamine promotes better stability of the polymeric chains and better wood panels' performance while minimizes the amount of free formaldehyde. However, due to melamine's high cost, its content has to be monitored until the desired specification. MUF resins have melamine content above 5 %, while melamine-fortified UF resins have up to 5 % [6].

2.2.3 Solids content

The solids content (SC) influences the quality and price of the resin and plays a crucial role in resin applicability, indicating the degree of polymerization. Depending on the percentage of nonvolatile content, the resin is suitable or not to produce a certain kind of wood panel. For example, the resin's SC must be around 60 % to 70 % to use as a binder for particleboard or fiberboard [7, 8].

2.3 Methods of analysis

The characterization of a UF resin can be done by different techniques and analytical tools. SC content is generally obtained by a prolonged process where a 2 g sample is dried at atmospheric pressure for 2 – 3 hours at 120 °C, while melamine content and U/F molar ratio can be measured by elemental analysis, HPLC, ^{13}C-NMR, FTIR, UV-VIS, LC-MS, and NIRS [5, 8].

2.3.1 Near-Infrared Spectroscopy (NIRS)

NIR covers the region between the visual spectra and the mid-infrared region with a wavelength range from 800 to 2500 nm (12500 to 4000 cm^{-1}). The most observed bands are the vibrations of –CH, –OH, –SH, and –NH bonds and are the results of overtones or combinations of each other. The absorption bands in the NIR region are wide, overlapping, and with a smaller amplitude than mid-infrared bands [9].

The most common molecular vibrations are stretching, bending, and rotational. Which one absorbs at a unique wavelength or frequency of the NIR spectrum depending on the bond type. These vibrations produce absorption bands, known as overtones, which generally appear between 780 and 2000 nm and result from the transitions between vibrational energy states. Regarding polyatomic molecules, the vibrational modes can interact with each other generating combination bands. These bands are the sum of each interacting frequency and usually appear between 1900 and 2500 nm [10, 11].

The most typical NIRS instruments have two basic configurations, as shown in Figure 3. They are based on transmittance or reflectance of light [12].

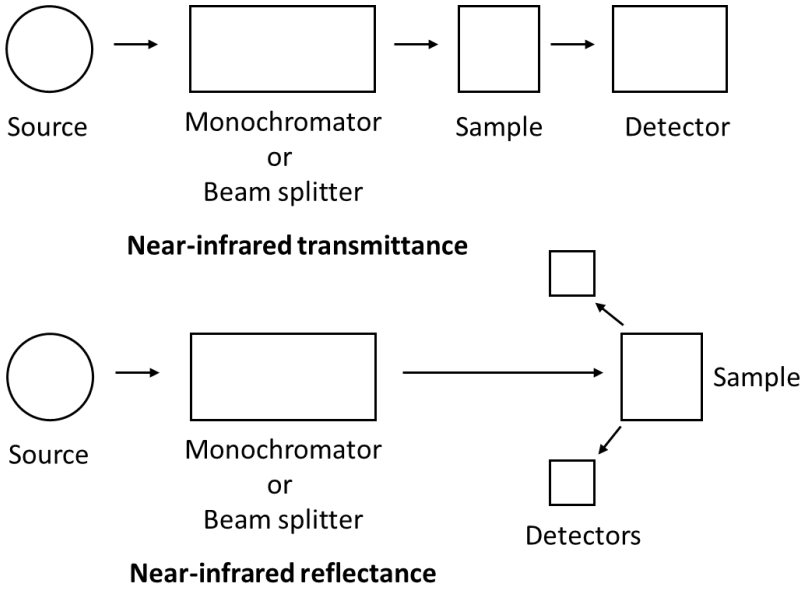


Figure 3 Most common NIRS instruments (adapted from [13]).

Both configurations have a source of light, a monochromator (or a beam splitter for Fourier transform spectrometers) to select the desired wavenumbers, a location for the sample, and a detector. As can be seen, the difference between them is the placement of the detector. The transmittance technique uses a detector in line with the sample, while the reflectance technique uses it between the monochromator and the sample to receive the reflecting light [12-14].

The transmittance technique has greater penetration, and the entire optical path is incorporated into the measurement to reduce the non-homogeneity of samples. This technique is suitable for measuring large particles due to their lower tendency to promote light scattering and loss of energy transmitted. While the reflectance technique is more suitable for fine particles since they are more likely to diffuse light [13].

NIRS can be combined with chemometrics to predict several properties of a sample. This combination is a usual indirect method that establishes a relationship between the absorption spectrum and different sample characteristics. It is a robust, fast, easy-to-use, non-destructive technique commonly used in quality and process control. However, the NIR spectra are complex and need calibration and preprocessing to predict the desired property with minimal error [15].

2.3.2 NIRS and chemometrics in amino resins

The NIR technique started to get more attention due to the increasing demand for product quality improvement and production rationalization in different fields, like chemical, petrochemical, pharmaceutical, food, and agriculture [16]. The NIR method emerged in substitution of time-consuming conservative analytical techniques (GC, HPLC, NMR, MS) and nonspecific control procedures (temperature, pressure, pH). Currently, with the development

of new technologies, NIRS is a powerful tool for industrial quality control and process monitoring, mainly in combination with light-fiber optics, new inline and online probe accessories, and chemometric evaluation. Moreover, there is no need for complex sample preparation, no waste production, reduced costs, and fast measurement [17, 18].

Chemometrics uses a combination of mathematics, statistics, and computer science to develop an indirect prediction model based on the chemical composition of a substance measured by different techniques. Chemometrics can be combined not only with NIR but also with NMR, XES, FT-IR, and HPLC [18].

Works using NIRS and chemometrics have been developed to predict the main properties of amino resins. Henriques *et al.*, in 2012, developed models to determine melamine content and F/U molar ratio using NIRS with partial least squares (PLS) regression [6, 19]. Gonçalves *et al.*, in 2020, published a model using NIRS for a rapid determination of SC with PLS regression [7, 12]. Also, they developed several models using PLS to predict the molar ratio of formaldehyde and urea, molar ratio of formaldehyde and melamine, molar ratio of formaldehyde and amino groups, total urea, and total melamine [12, 20].

The wavelength region between 4200 and 7500 cm^{-1} is the most used spectral zone to build a UF and MUF resin properties prediction models. Generally, due to the presence of water, this range can be divided into two, from 7502 to 6098 cm^{-1} and from 5000 to 4246 cm^{-1}, to exclude the water absorption peaks at 11800, 8600, 5620, 5200, 5150, and 4115cm^{-1}. The most significant vibration bands are presented in Table 1 with their specific characteristics [19, 21].

Table 1 Characteristics of the most significant bands presented in the UF resins spectrum (adapted from [21]).

Wavenumber (cm^{-1})	Characteristic	Wavenumber (cm^{-1})	Characteristic
7042 6920 6720	Triplet overtones of basic and complex NH vibrations	4926-4854	hydroxyl methylol groups vibrations range
5973 5834	Complex Methylene groups vibrations	4642 4549	NH combinations bands
5055 5030 4916 4878	Composite vibrations derived from amino and amide groups	4436	CH combinations bands

The MUF resin spectrum has a similar shape as the UF resin spectrum. The main difference appears in the region of 4651 and 5180 cm^{-1} which is connected to the presence of the s-triazine ring band of the melamine (located at 4789 cm^{-1}) [21].

2.4 Data Preprocessing

Preprocessing has an essential role in chemometrics modeling due to the improvement of the spectra quality without influencing the calibration model and removing valuable information. It promotes the development of a simpler and more robust model while removing physical phenomena like light scattering and noise, for example [10].

The most common and straightforward data preprocessing are the data mean-centering and variance scaling, which together originate the auto-scaling. The data mean-centering subtracts the mean spectrum ($\bar{x}$) of the data set from every spectrum sample, while variance scaling removes the standard deviation (s) only. Equation 1 represents the auto-scaling procedure made by the combination of the mentioned methods. It moves the data to the origin of the multivariate space and scales it to unit variance resulting in a simpler and interpretable regression model [22].

$$x_{corr} = \frac{x_{org} - \bar{x}}{s} \quad (1)$$

The others commonly used preprocessing methods can be divided into two main categories: scattering correction and spectral derivatives. Scatter correction minimizes the influence of nonlinearities and baseline shifts, while spectral derivatives decrease the noise by spectral smoothing [23].

2.4.1 Scattering correction

The principal preprocessing techniques of this category are Multiplicative Scatter Correction (MSC), Standard Normal Variate (SNV), and Normalization.

MSC is probably the most used NIR preprocessing technique. It consists of the estimation of the correction parameters (Equation 2) and the application of these coefficients in the spectrum correction (Equation 3) [23].

$$x_{org} = b_0 + b_{ref,1} x_{ref} + e \quad (2)$$

$$x_{corr} = \frac{x_{org} - b_0}{b_{ref,1}} = x_{ref} + \frac{e}{b_{ref,1}} \quad (3)$$

Where $\boldsymbol{x_{org}}$ is the measured spectrum to be corrected, $\boldsymbol{x_{ref}}$ is the reference spectrum, usually the average spectrum of the calibration set, $\boldsymbol{e}$ is the un-modeled part presented in $\boldsymbol{x_{org}}$, $\boldsymbol{x_{corr}}$

is the corrected spectrum, and $\boldsymbol{b_0}$ and $\boldsymbol{b_{ref,1}}$ are scalar parameters which are different for each sample. These parameters are found by the intercept ($\boldsymbol{b_0}$) and slope ($\boldsymbol{b_{ref,1}}$) of a plot of $\boldsymbol{x_{org}}$ and $\boldsymbol{x_{ref}}$ [23].

The second most applied NIR spectra preprocessing is the SNV, and it is similar to MSC because it uses the same format according to Equation 4. It consists of the subtraction of the mean value of the sample spectrum ($\boldsymbol{a_0}$) and scales it to unit variance, dividing it by the standard deviation of the sample spectrum ($\boldsymbol{a_1}$) [23].

$$x_{corr} = \frac{x_{org} - a_0}{a_1} \quad (4)$$

The normalization technique just removes the magnitude dividing $\boldsymbol{x_{org}}$ by the standard deviation ($\boldsymbol{a_1}$). The Equation used is similar to Equation 4, where $\boldsymbol{a_0}$ is equal to zero [23]. A derivative of the normalization preprocessing is the min-max normalization. This method, essentially found in the NIR Software (OPUS), used to develop some of the predictive models of this dissertation, scales the spectrum intensities to the effect that the minimum absorbance unit is 0 and the maximum is 2 [24].

Another method similar to SNV is the robust normal variate (RNV). The RNV is robust to outliers because it uses the interquartile range in the spectra correction. This method gives better results since outliers can often influence the sample mean/variance negatively. From Equation 5, the RNV method removes the mean and scales the data according to the interquartile range (IQR), which, for example, can be the difference between the 75th and 25th percentile [25].

$$x_{corr} = \frac{x_{org} - a_0}{IQR} \quad (5)$$

2.4.2 Spectral derivatives

Spectral derivatives techniques, as mentioned before, can decrease the noise presented in the spectra. The simplest pretreatments are based on the finite-difference between successive spectral points. It can be done by the first and second derivatives but should be avoided due to possible noise inflation [23].

The most common techniques of spectral derivatives use smoothing before the derivative to avoid inflation and the extreme decrease of the signal-to-noise ratio. These techniques are Norris-Williams derivation (NW) and Savitzky-Golay derivation (SG) [23].

NW firstly applies the smoothing by averaging over a given number of points (2m+1 window) centered around a measured point (i), as can be seen according to Equation 6.

$$x_{smooth,i} = \frac{\sum_{j=-m}^{m} x_{org,i+j}}{2m+1} \quad (6)$$

Where $x_{smooth,i}$ represents the smoothed spectral region, and m is the number of points above and below the measured point.

Then, the estimation of the first or second-order derivative can be done according to Equation 7 and Equation 8, respectively. Where a given gap, normally greater than two, is taken between two smoothed values.

$$x_i' = x_{smooth,i+gap} - x_{smooth,i-gap} \quad (7)$$

$$x_i'' = x_{smooth,i-gap} - 2.x_{smooth} + x_{smooth,i+gap} \quad (8)$$

The concept of the NW first derivative technique is presented in Figure 4.

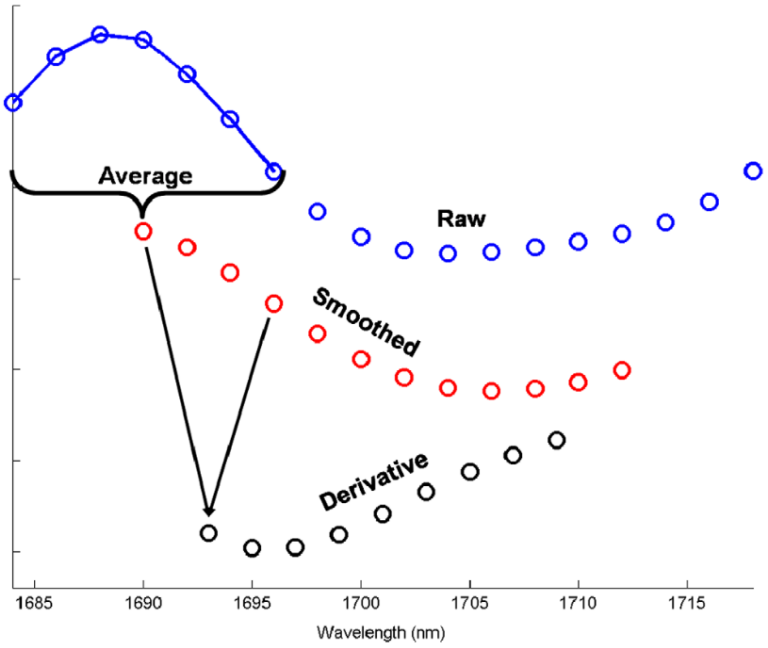


Figure 4 Norris-Williams first derivative concept (example of a seven-point window and a gap of three) [23].

The SG derivative creates a fitted curve, usually a third-order polynomial, on the raw data with a centered and defined symmetric window like the NW derivative. After calculating the polynomial parameters, it is easy to obtain the derivative of any order and create the corrected spectra [23].

An example of the application of the SG derivative is presented in Figure 5.

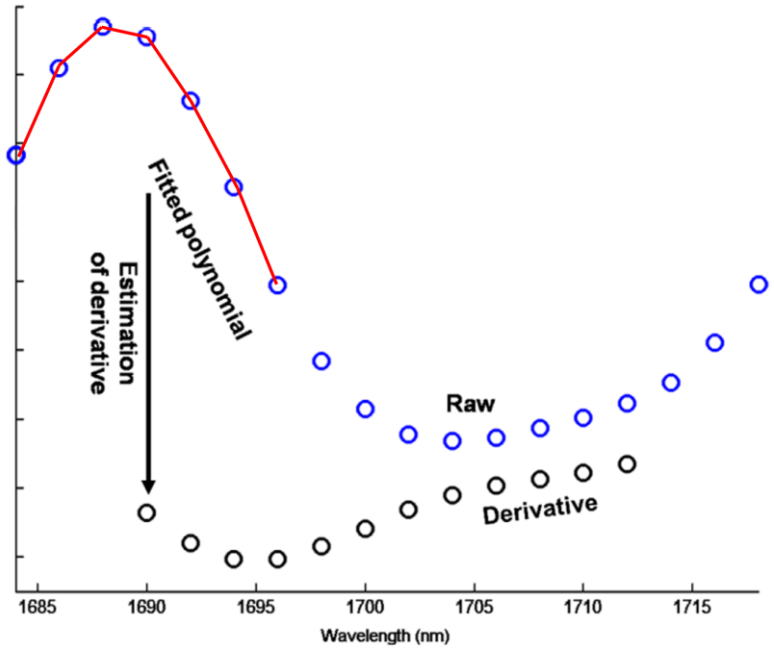


Figure 5 Estimation of the first derivative by SG (example of a 7-point window and a second-order polynomial) (adapted from [23]).

Both derivative techniques eliminate parts of the raw data, as noted in Figure 4 and Figure 5, at the beginning and end of the spectrum. Nevertheless, this issue is not crucial if the spectrum corresponds to a vector of more than 500 points; otherwise, loss of important wavelengths may occur [23].

Different combinations of correction methods are presented in the scientific literature, but developing many preprocessing steps is not advisable, and some initial rules must be followed [23]:

- Scatter correction should be done before differentiation.
- Normalization can be used in any step but doing it before any other operation it is easier to evaluate its effect.
- De-trending, which is a type of baseline correction, should be done after scattering correction and not as a first method.

2.4.3 Baseline correction

The most common baseline correction methods are the Constant Offset Elimination (COE) and Straight-line subtraction (SLS). The COE is one of the classical methods used to correct the baseline shift. Its simplest form comprises the removal of a constant value from each spectrum point, which can be a single point (minimum) or the average absorbance of points. The SLS is a more complicated way of offset correction. In this method, a straight line is calculated between two selected points, and the calculated values, based on this line, are subtracted from all corresponding points in the spectrum [26]. The line is calculated according to the Equation 9:

$$y(i) = \left[\frac{y_2 - y_1}{x_2 - x_1}\right] x(i) + \left[y_1 - \left(\frac{y_2 - y_1}{x_2 - x_1}\right) x_1\right] \quad (9)$$

Where (x_1, y_1) and (x_2, y_2) are the coordinates of the chosen points (wavenumber and absorbance), and $(x(i), y(i))$ are the points calculated.

2.5 Quantitative analysis

NIRS is an important tool for developing multivariate regression prediction models for quantitative analysis. It is necessary to use sufficient spectra, with at least 100 to 200 samples, to produce training sets to develop the calibration model and validation sets to analyze the model performance.

Multivariate regression is a technique employed to estimate a single property of a sample using more than one independent variable. It is commonly used in the development of methods in NIR quantitative analysis. The most used regression models are multiple linear regression (MLR), principal component regression (PCR), partial least squares regression (PLS), artificial neural network (ANN), and support vector machines regression (SVR) [9, 27].

2.5.1 Multiple linear Regression (MLR) and Principal Component Regression (PCR)

MLR is one of the oldest regression models, and due to the improvements in computation power, it is becoming outdated. It permits the usage of fewer wavelengths with more spectral representation and relates them with a defined sample property. The prediction property (y_j) can be described according to Equation 10 [28, 29].

$$y_j = b_0 + \sum_{i=1}^{k} b_i x_i + e_{i,j} \quad (10)$$

Where b_0 and b_i are the computed coefficients, x_i is the absorbance at each wavelength and $e_{i,j}$ is the error.

The MLR study is done based on each spectra wavelength and its predictive ability. The approach is effective when a small wavelength number is used and in a simple system with low degrees of freedom. Thus, this method is not advisable for samples with complex chemical compositions and different physical conditions [28, 29].

PCR is an alternative to MLR with many advantages over it. Due to the presence of a large number of independent variables, commonly encountered in multiple regression analysis, PCR becomes a great method to avoid the problems induced by collinearity or correlation between variables. These problems are related to the use of unnecessary variables that add redundancy to the model and decrease predictive performance [29].

PCR is divided into two steps. Firstly, the mean-centered NIR spectra are treated with principal component analysis (PCA) to summarize the raw data. Then, a regression process is done to relate the scores to the y variables [29].

PCA promotes spectral decomposition after mean centering by reducing the dimension of the raw data and using the maximum variations between the spectra. The new dataset represents the training set and comprises two different matrices: the eigenvectors and the scores, both needed to reconstruct the raw data by a linear combination [29]. PCA creates a set of orthogonal vectors called principal components, which are defined based on a linear combination between the original variables and the amount of explained variance in component directions [30].

Using the PCA algorithm, the Equation 11 is obtained with the selected principal components:

$$A = SV + E_A \quad (11)$$

Where A is the raw data represented by the spectral absorption matrix (n by w), S is the scores values matrix (n by m), V is the eigenvector matrix (m by w) and E_A is the residual spectra matrix (n by w). Each of these matrices has a proper dimension where n is the number of spectra, w is the number of wavelengths, and m is the number of principal component eigenvectors [29, 31].

After calculating the scores from each spectrum in the S matrix, it is possible to obtain the regression model for the concentrations (properties), as shown in Equation 12.

$$Y = CS^T + E_Y \quad (12)$$

Where Y is the properties matrix (p x n), C is the regression coefficients matrix (p x m), E_Y is the error matrix (p x n), p is the number of constituents for calibration, and T indicates the transpose of the matrix [29, 31].

The coefficients matrix is obtained according to Equation 13:

$$C = YS(S^TS)^{-1} \quad (13)$$

After the elimination of the noise error matrix presented in Equation 11, the S can be obtained by Equation 14, considering V as an orthogonal matrix in which V^{-1} is equal to V^T .

$$S = AV^{-1} = AV^T \quad (14)$$

Then, the PCR formula (Equation 15) is obtained by the combination of Equation 12 and Equation 14.

$$Y = CVA^T + E_Y \quad (15)$$

2.5.2 Partial Least Squares (PLS) Regression

The PLS regression method was created in response to PCR to avoid the hard selection of principal components but with the same objective of summarizing the original data. In this case, PLS uses factors obtained by maximizing the covariance between the x and y variables. In other words, unlike PCR, PLS regression uses both the mean-centered spectral data, X matrix, and the y properties in the estimation [29].

The PLS regression methodology starts with the linearization of the spectral data (X), according to Equation 16, transforming it into scores (S) and loadings (P), similar to PCA. In parallel, the same is done with the y variables according to Equation 17, where U and q are the y scores matrix and y loading vector, respectively, and E and e are the residuals [29].

$$X = SP^T + E \quad (16)$$

$$Y = Uq^T + e \quad (17)$$

After, the objective of the PLS algorithm is to find the best regression coefficient in the linear relation between X and y. For this, the components, or latent variables, are calculated for both blocks (X and y) with the best relation between them. In other words, with the maximum covariance between X and y scores. Moreover, the components are selected based on three criteria: high correlation with y, high variability with X, and no components correlation [32, 33].

The final model is presented in Equation 18, where B is the coefficient matrix, and F is the residual error.

$$X = BY + F \quad (18)$$

2.5.2.1 Variable selection methods with Partial Least Squares

Different PLS derivatives methodologies were created to develop models with minimal irrelevant and noise information. The objective of these methodologies is to remove variables, wavelengths, related to information that can influence the model negatively. The selection of a proper wavelength region promotes the development of a more reliable and efficient model. Some of those methods are interval Partial Least Squares (iPLS), Forward interval Partial Least Squares (fiPLS), and Synergy interval Partial Least Squares (siPLS).

The iPLS comprises the analysis of equidistant portions of the full spectrum. The PLS methodology is applied in each wavelength portion, and the best region is selected according to its prediction performance. By using the same number of latent variables, the regions are evaluated according to the values of the root mean squared error (RMSE) or the coefficient of determination (R^2) of cross-validation (CV) and compared with the PLS model applied in the entire spectra [34].

The fiPLS is an extension of the iPLS model. The difference between them relies on the number of intervals selected. While iPLS selects only one interval, fiPLS promotes the combination of 2 intervals to create a model with minimal RMSECV or maximum R^2CV. This methodology starts with iPLS to choose the first interval. After, the remaining ones are combined with it until no further improvement is possible [35].

siPLS allows the combination of more intervals to create a model with better predictability. The siPLS method tries different combinations of intervals and selects the best one by evaluating the RMSECV and R^2CV [36].

2.5.3 Artificial Neural Network (ANN)

ANN has a major advantage in developing prediction models when there is a non-linear relationship between the x and y variables. The typical ANN arrangement structure used in NIR prediction models is the multilayered perceptron (MLP) network, a feed-forward ANN commonly used with a backpropagation algorithm. This structure comprises several layers divided into one or more neurons (nodes) responsible for the connection between layers and the transmission of information needed for the prediction [9, 37].

The MLP structure is composed of three types of layers. The input data, also called the input layer, represents the x variables, while the output data, or output layer, represents the y variables. The hidden layers, which represent the modeling process, are placed between the inputs and the outputs according to Figure 6 [37, 38].

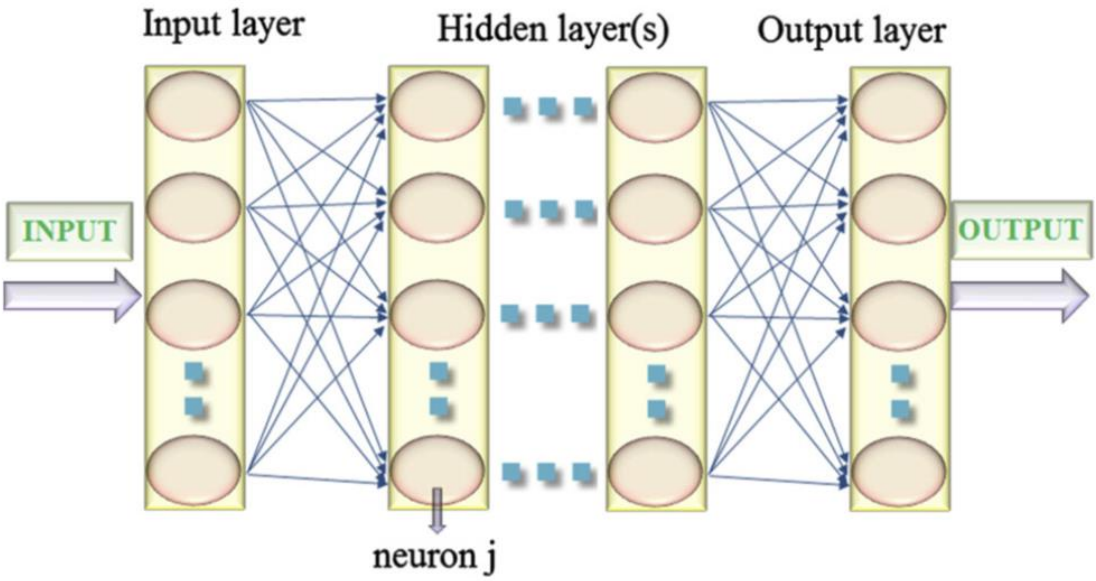


Figure 6 Typical ANN arrangement structure for NIR prediction models [37].

A neuron (j) is responsible for the transmission of information (I_j) of the input data (X_i) through the ANN structure. The connections between layers are made by connection weights ($W_{i,j}$), blue

arrows in Figure 4. They are calculated, along with the bias term (θ_j), according to the mapping capability of the trained network [37]. The following Equation 19 represents how these terms are related to each other:

$$I_j = \sum_{i=1}^{n} X_i W_{i,j} + \theta_j \quad (19)$$

The network training is done by backpropagation, a frequently used training algorithm, according to the transfer function selected in the hidden layers and by the desired variables outputs of the calibration set. The objective of the training is to calculate the best values of the weights and bias terms to predict the properties efficiently [37]. These parameters are optimized by a solver function which can be 'lbfgs', an optimizer in the family of quasi-Newton methods, 'sgd', which refers to stochastic gradient descent, or 'adam', which refers to a stochastic gradient-based optimizer [39]–[41].

The most used transfer function, or activation function, is the logistic sigmoid, Equation 20, which is responsible for the prediction of the y_j property,[42].

$$y_j = f(I_j) = \frac{1}{1 + e^{-I_j}} \quad (20)$$

However, ANN supports a wide range of transfer functions such as linear, hyperbolic tangent, sign, and step [43].

The optimization of an ANN can be challenging due to the diverse associated parameters such as the number of hidden layers, number of neurons, and type of the activation and solver functions. A good approach for parameters selection is to start with a low number of hidden layers and neurons with specific activation and solver functions. Then, increase the neuron number until the desired ANN performance is reached [44]. However, it is important to mention that the number of neurons affects the predictive capability of the ANN. Too many neurons decrease the robustness of the model; in other words, it reduces the capacity to generalize the model for a different data set. While too few neurons affect the learning process making it difficult to determine the best values for the predicted properties using less accurate weights and bias terms values [37, 45].

2.5.4 Support Vector Machines (SVM) Regression

The SVM was initially created as a binary classification method and afterward adapted for regression. Originally the method consisted of a category differentiation technique. The input samples (training set) represented by vectors in a mapped space are classified by maximizing the gap between categories. After, the test or the cross-validation set is mapped into that same space and sorted to validate the model. The category prediction is made according to which side of the mapped space the validation set falls [46, 47].

The SVR model consists of finding the line, or hyperplane in a higher dimension, that best holds the training observations. In other words, the goal of the model is to find the function that best fits the training set without surpassing the margin ε (decision boundaries) so much, as shown in Figure 7 [48].

The support vectors are the data points closest to the margin, inside or outside of it. They tell the shape of the hyperplane and define the distance between the hyperplane and the up and lower decision boundaries [48].

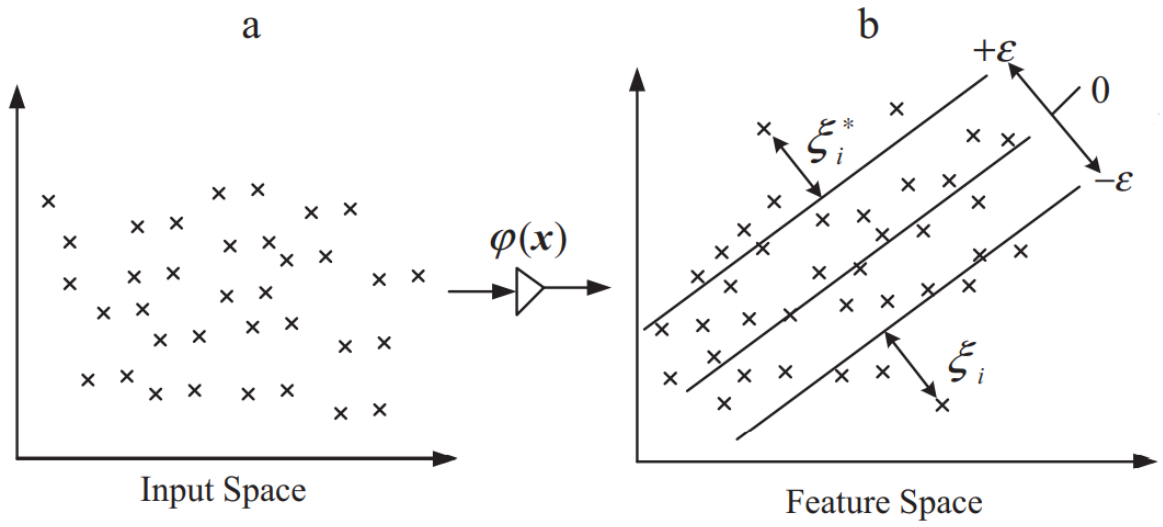


Figure 7: SVR transformation process. A nonlinear mapping function φ(x) convert the original data (a) to a linear problem in a feature space (b)[49].

According to Figure 7 and considering a training set consisting of n observations ($\{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^d \times \mathbb{R}$). The SVR transformation starts with the conversion of the original data by $\varphi(x)$. The nonlinear function $\varphi(x)$ is composed of a kernel function responsible for the projection of the original data into the feature space where all the data can be linearly presented. The commonly used kernel functions are linear, polynomial, and radial basis (RBF) [48, 49].

As early mentioned, the objective of the SVR is to find the function that best represents the relationship between x and y variables within the margin ε and as flat as possible. The linear relationship is represented by Equation 21, where $< \beta, x >$ depicts the dot product between x and β weights(gains) with $c_0 \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$.

$$f(x) = < \beta, x > + c_0 \qquad (21)$$

The improvement of the model comprises the maximization of the margin to hold the maximum training observations possible. Therefore, a convex optimization problem is done by

$$\text{Minimize } \frac{1}{2}\|\beta\|^2$$
$$\text{Subject to } \begin{cases} f(x_i) - y_i \leq \varepsilon \\ y_i - f(x_i) \leq \varepsilon \end{cases} \quad (22)$$

This optimization procedure ensures the *flatness* of the model by minimization of the norm ($\|\beta\|^2 =< \beta, \beta >$). The flatness contributes to predictions less sensitive to perturbations and allows finding a hyperplane that holds all the training data points without deviations larger than ε [48].

As can be seen from Figure 7, due to the presence of slack variables located outside the margin ε with a distance ξ^* from the upper boundary and ξ from the lower boundary, it is hard to optimize the model with such a restricted margin without considering them. Thus, positive slack variables are introduced to the optimization problem to allow for some errors and cope with infeasible constraints (Equation 23) [48].

$$\text{Minimize } \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$
$$\text{Subject to } \begin{cases} f(x_i) - y_i \leq \varepsilon + \xi_i^* \\ y_i - f(x_i) \leq \varepsilon + \xi_i \\ \xi_i, \xi_i^* \geq 0 \end{cases} \quad (23)$$

The regularization parameter (C) determines the tolerance of the model regarding the points outside the margin ε and helps to prevent overfitting. It represents the balance between *flatness* (model complexity) and the number of points with deviations more than ε [48]. The values of C and ε are problem and data dependents defined by the user [47].

3 Materials and Methods

The procedure for developing the models is summarized in Figure 8, and each step is explained in detail further ahead.

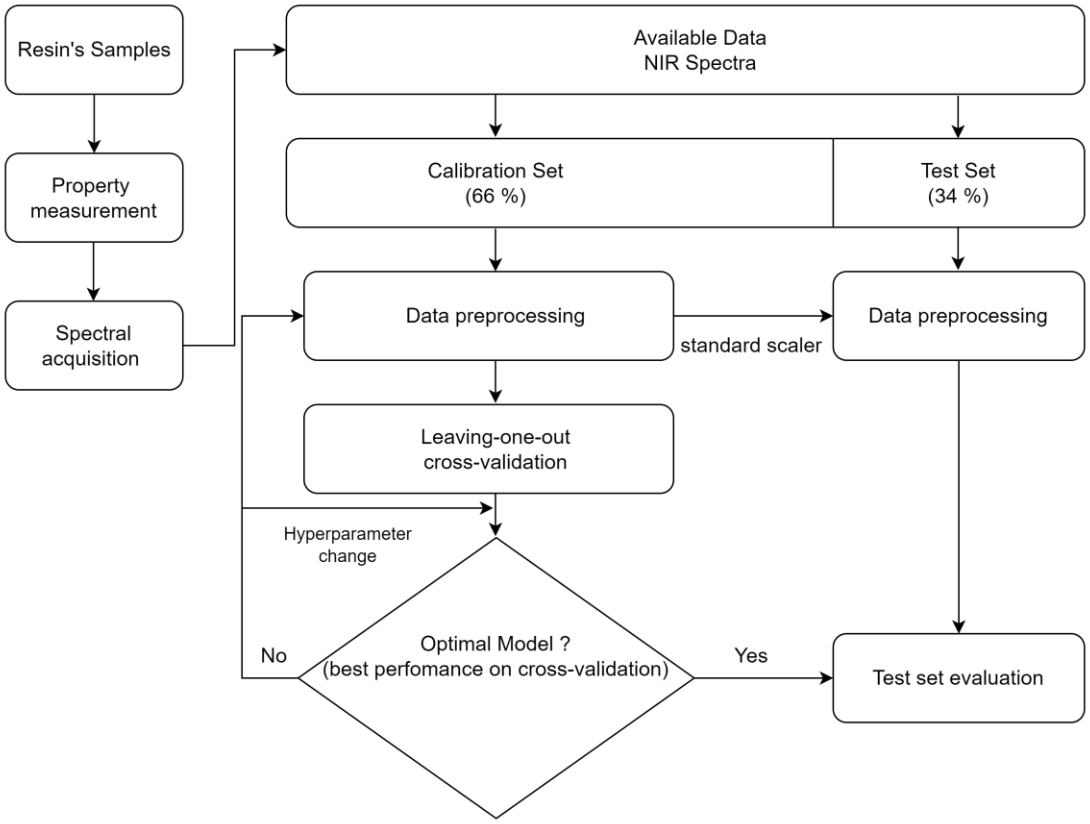


Figure 8 Summary of the models' development procedure.

The procedure starts with measuring the main properties, followed by the spectral acquisition of all samples. Next, the database is split into calibration and test sets to create and evaluate the models.

The model creation strategy relies on selecting the preprocessing method, including the spectral region, with the best cross-validation performance. After the data split, the same preprocessing method is applied to both datasets, and the calibration set's mean and standard deviation are computed to scale the test set. Then, the cross-validation approach is performed to select the best model. If the optimal model is achieved, the test evaluation is implemented to confirm its efficiency regarding an independent data set. If it is not, another preprocessing method is analyzed or a different hyperparameter (for example, number of latent variables, number of neurons, or value of the regularization parameter).

3.1 Reference methods

The main resin properties were measured according to some procedures already related in the scientific literature. These measures were essential to use as reference values and evaluate the models' performance.

The reference for the formaldehyde urea molar ratio (FUMR) was based on the manufacturer information, while melamine and solids contents were acquired by methodologies commonly used in the company.

The FUMR is related to the amount of formaldehyde and urea used in resin production. It is calculated based on the quantity of these reagents in the synthesis process.

The melamine content of the samples was measured using a UV-VIS spectroscopic method in which the melamine ion is identified at a specific wavelength (234 nm). To form the melamine ion, a small amount of the resin, around 200 mg, is hydrolyzed in a round bottom flask by reflux with 50 mL of 0.1 M of Hydrochloric acid (HCl) for 2 hours. After, the flask content is poured into a 100 mL volumetric flask and diluted with HCl 0.1 M. Next, two parts of the volumetric flask, one of 5 mL and the other of 10 mL, are diluted with HCl in 50 mL volumetric flasks. Their melamine contents are measured by the UV-VIS spectrophotometer, and their average value is used considering the dilutions made. The result is a percentage (w/w) of melamine in the resin.

Solids' content was measured in triplicate by a dry process in which a low amount of resin is set in an oven. The method commonly used by amino resins manufacturers have the following steps:

1) Weight an aluminum capsule (w_c).
2) Add around 2 g of resin into the capsule and weight the set capsule + resin (w_{cr}).
3) Bring the capsule to a ventilated oven for 3 hours at 120 ˚C to evaporate the volatile content.
4) After drying, weight the set (capsule + solid) (w_{cs}).
5) Calculate the SC with Equation 24:

$$SC(\%w/w) = \frac{w_{cs} - w_c}{w_{cr} - w_c} \quad (24)$$

3.2 Spectral acquisition and software

Spectra of each sample were acquired in transmittance mode and converted to absorbance. The acquisition was made in triplicate, in the range of 4000 cm^{-1} to 12000 cm^{-1}, and using air as a background spectrum. Before measurements, the samples were kept at 25 °C in a thermostatic bath to maintain a standard condition. Spectra acquisition was made in an average of 32 scans each with a resolution of 8 cm^{-1}, and it took about 30 s of scanning time. The complete spectrum comprises 2074 data points but can be decreased when essential bands of the resin spectrum are selected to create a more reliable and efficient model. A total of 729 spectra were obtained, which correspond to 243 samples.

The spectrophotometer was a Fourier Transform – NIR instrument Matrix-F from Bruker Optics with a transmittance probe from Solvias (Safiro 12S-150). The software used for the spectra measurement was OPUS (6.5 version) by Bruker, which also treated the spectra to create models based on PLS. At the same time, models based on PLS, ANN, and SVR were created in Python using an open-source machine learning library (*Scikit-learn*) and a semi-automated preprocessing model for NIRS data (NIPPY) to analyze different pretreatment spectrum combinations [50].

3.3 Validation methods

The evaluation procedure was made in two different strategies: cross-validation and test validation.

Firstly, the data set was split into two groups: 66 % was used for calibration (482 spectra), while 34 % was set aside as a test set for final validation of the selected model (247 spectra).

The type of cross-validation carried out was the leaving-one-out in which an individual sample is taken out from the calibration set, and the remaining ones are used as the new training set. The procedure is made according to the following steps:

1) Removal of spectrum number 1 from the calibration set;

2) Creation of the new calibration set from the remaining samples;

3) Property prediction of spectrum number 1;

4) Spectrum 1 is replaced in the calibration set, and another model with a different calibration set is created with the removal of spectrum 2;

5) Every step is repeated until all 482 spectra are removed each time and all properties predicted.

The performance of the regression methods was evaluated according to the coefficient of determination (R^2) and Root Mean Squared Error (RMSE), which expressions are presented in Equation 25 and Equation 26, respectively.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y}_i)^2} \quad (25)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \quad (26)$$

Where N is the number of calibration or validation spectra; i is the sample number; $\hat{y}_i$ is the predicted value; y_i is the reference value; $\overline{y}_i$ is the average of the reference values.

3.4 Models Evaluation

The evaluation of the models was done with different approaches according to the parameters related to each model.

The PLS models created in OPUS and Python were analyzed regarding the number of latent variables (LVs) for different preprocessing methods. The best LV number was selected when the minimum RMSE, or maximum R^2, of cross-validation was reached among the 15 studied LVs.

The study of the ANN models was done, as a first approach, by investigating the activation and optimizer functions. Using a simpler ANN composed of one input layer, one output layer, and a unique hidden layer with up to 30 neurons, the study was performed after mean centering, unit variance scaling, and PCA to decrease the dimension of the x variables and the computing time. The evaluation was done using the RMSE and R^2 to determine the best functions and the number of neurons.

The MLP regressor model used to develop the ANN is presented in the Scikit-learn library with four options for the activation function, 'identity' (no activation function); 'logistic' (sigmoid); 'tanh' (hyperbolic tangent), and 'relu' (rectified linear unit), and three for the solver (optimizer) function ('lbfgs'; 'sgd' and 'adam').

Three SVR parameters, also presented in the Scikit-learn library, were analyzed to develop the Support vector machines models:

- Kernel function, two options were studied ('linear' and 'RBF').
- Regularization parameter (C), with values in the range of hundreds or thousands.
- Epsilon (margin ε), minimum possible to restrict the predicted values to be closer to the reference values.

Like the previous ones, the SVR model was also evaluated by the RMSE and R^2 in order to choose the best parameters.

It is important to mention that the evaluation of the models considered not only the best performance regarding the root mean squared error, or the coefficient of determination, of cross-validation (RMSECV or R^2CV, respectively), but also the adjustment of the model to the calibration set (RMSECal or R^2Cal) compared to the test set. The common problem is finding the models' best parameters to balance underfitting and overfitting. It is visualized by the difference between the RMSE (or R^2) of calibration and test set validation. The overfitting occurs when the model is very data-dependent on the calibration set and gives poor prediction results. The RMSETest has a notably larger value than the RMSECal (or the opposite behavior for the R^2). The underfitting occurs when the model is not large enough or has inadequate parameters to capture the significant variability in the data. It appears when the RMSETest is smaller than the RMSECal (or the opposite behavior for the R^2) [51].

The Figure 9 represents well what happens when the model overfits or underfits, for PLS models, for example.

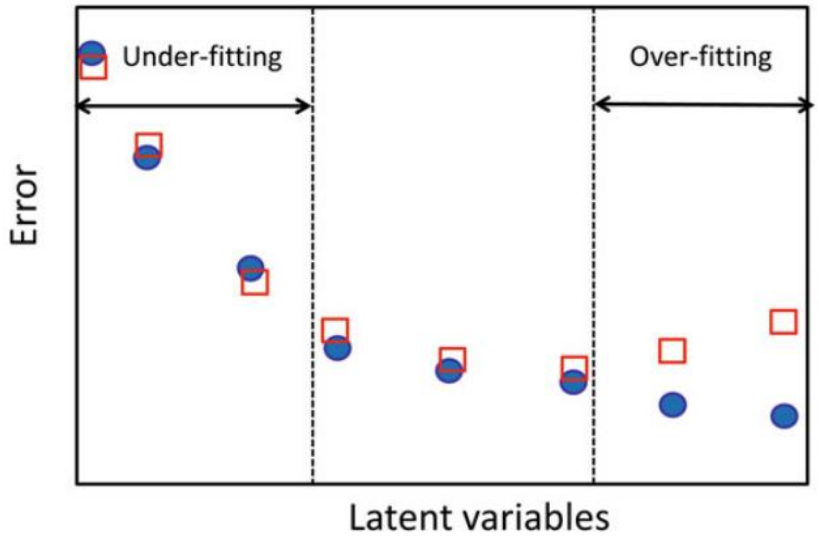


Figure 9 Changes in prediction error for calibration samples (blue circles) and independent samples (red squares) as a function of the number of LVs [51].

4 Results and Discussion

The objective of this dissertation was to develop different NIR models based on PLS, ANN, and SVR, to predict the principal resin's properties (FUMR, melamine, and solids contents). The primary approach was to evaluate the models using the main regions of the spectrum and optimize them by modifying their different parameters. For PLS, using many preprocessing methods, the optimization was done by using the best number of LVs, for ANN by the number of neurons in the hidden layer, and for SVR by the parameters C and Epsilon.

The best models are presented in this section with their main parameters, preprocessing, evaluation performance, and spectral zone.

4.1 Wavenumbers Selection

The selection of the wavenumbers was based on previous works related to amino resins and presented in the scientific literature [19, 21, 52]. Figure 10 represents the spectra of the 243 samples where the numbers highlight the prominent bands.

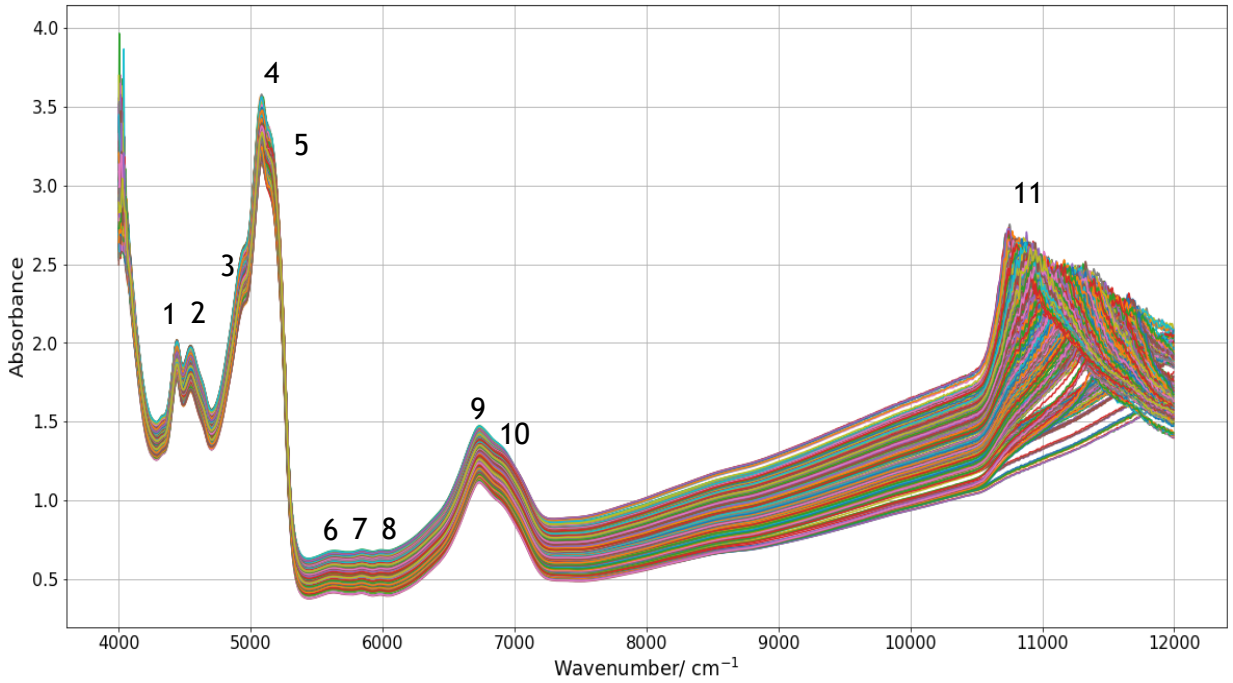


Figure 10 Spectra of the UF resins used in the development of the models

As shown in Figure 10, the range from 4200 and 7500 cm^{-1} is commonly used for developing NIR models in amino resins, as described by Kasprzyk *et al.* and Henriques *et al.* [19, 21]. It comprises almost all the bands presented in the spectrum except the band number 11, related to the third overtone of the CH bond, which can be excluded due to the excess noise [53]. Also, the region between 4800 and 5200 cm^{-1} can be removed due to water absorbance in a large and broad band, a common problem in NIR analysis [19].

As a detailed description of the important bands of the resins' spectra, band number 1 at 4440 cm^{-1} is related to the CH stretching and CH bending combination commonly presented in the formaldehyde NIR spectrum. Band number 2, at 4544 cm^{-1}, is often shown in the urea NIR spectrum and is related to the NH symmetrical stretching and NH bending combination. This band can also overlap the band related to melamine, at around 4700 cm^{-1}, mainly in melamine-fortified UF resins in which the amount of urea is more significant than the melamine [21, 53]. Band number 3 at 4950 cm^{-1} represents the second overtone of the carbonyl group (C=O) associated with urea, while band number 4 at 5087 cm^{-1} represents the NH stretching (asymmetric) and NH in-plane bending combination also related to urea and presented in pure melamine spectrum. Band number 5, which seems from 5000 to 5200 cm^{-1}, represents the OH stretching and HOH band combination associated with the water molecule. Bands 6 (5633 cm^{-1}), 7 (5844 cm^{-1}), and 8 (5987 cm^{-1}) are associated with the first overtone of CH from methylene (CH_2), CH associated with OH, and CH associated with carbonyl from formaldehyde, respectively [53, 54]. Band 9 at 6730 cm^{-1} is related to the first overtone of NH stretching from urea, amides, and secondary amines. Band number 10 at 6890 cm^{-1} is associated with the carbonyl group's third overtone and OH bond's first overtone from primary or polymeric alcohols. Also, this band overlaps the one related to melamine which appears around 6900-6700 cm^{-1} attributed to $ArNH_2$ [55]. The broad band that seems from 6240 to 7100 cm^{-1} is associated with the OH stretching from water and alcohols [53].

The models' creation focused on the most representative spectral regions, in this case, on three principal regions:

- From 4100 to 7900 cm^{-1}, as a first approach, to include the main bands and compare the improvement or not of the models after choosing more restricted regions.
- A combination of two regions (4300 - 4800 cm^{-1} and 5400 - 7500 cm^{-1}) related to bands 1 - 2 and 6 - 10, after eliminating the water region, which has a greater absorbance and can negatively influence the model.
- From 5400 to 7500 cm^{-1}, which is a region more related to the polymer chain.

4.2 PLS models in OPUS

The OPUS software was an essential tool for developing the models due to its user-friendly interface, easy manipulation, and application of the models. Since it is a standard software already used by the company, the models developed can be promptly applied.

The PLS models were created using different spectral preprocessing available in the software. For each model, ten preprocessing methods were applied to the regions mentioned before, with mean-centering of the spectra (x variables) and the properties (y variables). The preprocessing methods were COE, SLS, SNV, Min-Max normalization, MSC, First derivative,

Second derivative, First derivative + SLS, First derivative + SNV, and First derivative + MSC. The more distinguished ones, with more performance, are presented.

The analysis of the models performed in the different spectral regions was done by leaving-one-out cross-validation to choose the best LV and preprocessing method. After, a test validation was done with the best-performing one to evaluate the model regarding an independent data set.

4.2.1 FUMR model – OPUS

The statistical summary of the best FUMR models created in OPUS is presented in Table 2 for a specific region (from 5400 to 7500 cm^{-1}). Studies regarding the other spectral regions are presented in Annex A and can be compared. All the models achieved good calibration and validation performance with high R^2 and low RMSE. It shows that all the main regions of the resins' spectra can be used to develop the FUMR model, but some can be distinguished to create the one with better performance.

Table 2 Summary of the OPUS - PLS models for FUMR using the wavenumber range 5400-7500 cm^{-1}

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	LV
-	0.9930	0.00252	0.9898	0.00300	
SLS	0.9941	0.00233	0.9906	0.00288	
COE	0.9936	0.0242	0.9903	0.00293	15
1st derivative + SNV	0.9941	0.00233	0.9901	0.00296	
1st derivative + MSC	0.9937	0.00240	0.9894	0.00306	

According to Table 2, the best preprocessing method is the SLS, providing the lowest RMSECV and highest R^2CV among the three studied spectral zones. The region related to the polymeric chain appears to be the most suitable for predicting the FUMR. The comparison of the regions exhibited the difference in the performance of the developed models and confirmed that the region related to the greater water absorbance influences the model negatively. Since the models that include this region had the worst performance.

After selecting the best model, the final test validation was implemented. The R^2Test and RMSETest also had good values with 0.9895 and 0.00266, respectively.

4.2.2 Melamine content model – OPUS

According to Table 3, the melamine content models showed the best performance in the combination region of 4300 - 4800 with 5400 - 7500 cm^{-1}, where the water band with higher absorption is excluded. The study of the other spectral zones is also presented in Annex A for

comparison. The combination region seems to be the best one to develop this model since it is the spectral zones related to the melamine molecule in which the NH stretching bands are presented. Also, it shows that the melamine models do not perform so well since it is presented in a low amount, and the bands of the resin's principal components overlap melamine's bands.

Table 3 Summary of the OPUS - PLS models for Melamine content using the wavenumber ranges 4300-4800 and 5400-7500 cm^{-1}

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	LV
-	0.8500	0.0795	0.7824	0.0942	
COE	0.8520	0.0790	0.7830	0.0941	
Min-Max Norm.	0.8515	0.0791	0.7803	0.0946	15
SNV	0.8547	0.0783	0.7783	0.0951	
SLS	0.8618	0.0763	0.7725	0.0963	

The melamine models did not perform as well as the FUMR but have a good predictive ability. The prediction error is close to the obtained by the reference method, with a maximum around 0.2 %. The COE is the best preprocessing method, with the lowest RMSECV and highest R^2CV. Regarding test validation, the R^2Test and RMSETest are 0.8307 and 0.0983, respectively.

4.2.3 Solids content model – OPUS

The SC model performed similarly to the FUMR model for the same spectral regions. The regions' comparison can be made based on Table 4 and the results presented in Annex A. As can be seen, the SC models have a good performance regarding all main spectral zones, with R^2CV higher than 0.96. Moreover, its improvement occurs with the water band elimination also done in FUMR and melamine models.

Table 4 Summary of the OPUS - PLS models for Solids content using the wavenumber range 5400-7500 cm^{-1}

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	LV
-	0.9842	0.1010	0.9769	0.1200	
MSC	0.9854	0.0967	0.9765	0.1210	
SLS	0.9853	0.0971	0.9765	0.1210	15
Min-Max Norm.	0.9850	0.0982	0.9765	0.1210	
SNV	0.9847	0.0992	0.9765	0.1210	

According to Table 4, the best-performing model is without preprocessing for the region between 5400 e 7500 cm^{-1}. The region related to the polymeric chain seems more suitable for predicting the resin's solids content. This region appears to represent the relationship between volatile and nonvolatile compounds at 120 ºC, where we can find bands of formaldehyde and water as volatiles and bands related to methylolureas and methylolmelamines, which are dehydrated during the dry process and form non-volatiles compounds like the resin's polymer.

Regarding test validation, the R^2Test and RMSETest also had good results with values of 0.9770 and 0.1180, respectively.

4.3 PLS models in Python

Since preprocessing is an essential step in improving the multivariate models and is often determined through a trial-and-error procedure. The development of the PLS models in Python was based on an open-source tool called NIPPY to automate the preprocessing of the NIR data and compare different combinations [48]. Before the standardization of the spectra and properties, by removing the mean and scaling to unit variance, four preprocessing categories presented in the platform were used: scatter correction, smoothing, derivatives, and trimming. The configuration parameters of the module are presented in Table 5.

Table 5 Configuration parameters for the PLS models

Preprocessing operation	Parameter
Trimming	bins = 4100-7900; 4300-4800, 5400-7500; 5400-7500
SG	Filter window = 7, 11, 13 Polynomial order = 3 Derivative order = 0, 1, 2 also_skip=True*
MSC	also_skip=True*
SNV	also_skip=True*
RNV	IQR = 75, 25; 90, 10 also_skip=True*
Baseline (Mean-centering)	-
Normalization	also_skip =True*

*also_skip – a Boolean that generates a pipeline where the preprocessing operation is left out.

A total of 300 combinations of the preprocessing techniques presented in Table 5 were created (100 for each selected region). The study was done on each combination through the number of latent variables and based on the RMSE and R^2 of calibration, cross-validation, and test. The code presented in Annex B allows the automatization of the analysis saving in an excel file, created by the user, the results regarding each combination. It saves which preprocessing or combination was done and their results regarding the RMSE, R^2, and LV, based on the lowest RMSECV and highest R^2CV (Figure 11).

	A	B	C	D	E	F	G	H	I
1	Pre-processing	RMSECal	RMSECV	RMSETest	R2Cal	R2CV	R2Test	lv	Combination number
2	1;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': {'iqr': [75.0, 2	0.00266	0.004148	0.00388	0.991988	0.980512	0.977719	15	1
3	2;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': None, 'SAVG(	0.002832	0.003349	0.009752	0.990914	0.987297	0.859249	15	2
4	3;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': None, 'SAVG(	0.002999	0.003407	0.008287	0.989814	0.986853	0.898367	15	3
5	4;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': {'iqr': [75.0, 2	0.003038	0.005756	0.004939	0.989544	0.96247	0.963897	15	4
6	5;{'BASELINE': None, 'MSC': None, 'NORML': None, 'RNV': {'iqr': [75	0.003038	0.005756	0.004942	0.989544	0.96247	0.963852	15	5
7	6;{'BASELINE': None, 'MSC': {}, 'NORML': {}, 'RNV': None, 'SAVGOL':	0.002998	0.003443	0.00372	0.98982	0.986571	0.979515	15	6
8	7;{'BASELINE': None, 'MSC': None, 'NORML': None, 'RNV': None, 'SA	0.002956	0.003393	0.003387	0.990101	0.98696	0.983026	15	7
9	8;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': {'iqr': [90.0, 1	0.002664	0.004138	0.004003	0.991959	0.980607	0.97628	15	8
10	9;{'BASELINE': {}, 'MSC': None, 'NORML': {}, 'RNV': None, 'SAVGOL':	0.001963	0.002936	0.006114	0.995634	0.990238	0.944674	15	9
11	10;{'BASELINE': None, 'MSC': {}, 'NORML': {}, 'RNV': None, 'SAVGOL'	0.00256	0.003962	0.004031	0.992576	0.982215	0.975947	15	10
12	11;{'BASELINE': None, 'MSC': None, 'NORML': None, 'RNV': {'iqr': [7	0.003527	0.003964	0.004009	0.985911	0.982201	0.976216	15	11
13	12;{'BASELINE': {}, 'MSC': None, 'NORML': None, 'RNV': None, 'SAVG	0.003114	0.003482	0.003469	0.989016	0.986265	0.982195	15	12
14	13;{'BASELINE': {}, 'MSC': None, 'NORML': {}, 'RNV': None, 'SAVGOL'	0.002006	0.003072	0.005809	0.995442	0.989311	0.950055	15	13
15	14;{'BASELINE': {}, 'MSC': None, 'NORML': None, 'RNV': None, 'SAVG	0.002006	0.003072	0.003005	0.995442	0.989311	0.986635	15	14
16	15;{'BASELINE': None, 'MSC': None, 'NORML': {}, 'RNV': {'iqr': [75.0,	0.002308	0.003304	0.003329	0.993969	0.987634	0.983596	15	15

Figure 11 Arrangement of the preprocessing results saved in an excel sheet during the development of the PLS models in Python.

Plots were created for the RMSECV and R^2CV (presented in Annex C) to select the best combination and easier visualization. Each number, from 1 to 300, corresponds to a specific preprocessing combination created by NIPPY and saved in the excel file. After selecting the best preprocessing, the RMSE and R^2 of calibration, cross-validation, and test were studied to visualize the model's fitting. It can be seen from the figures also presented in Annex C.

4.3.1 FUMR model - Python - PLS

Figure 12 represents all the models created to predict the FUMR with the lowest RMSECV. It can be seen that all the three studied regions have good performance, as also the FUMR models developed in OPUS. According to the RMSECV and R^2CV plots, the best model is presented in the combination region of 4300-4800 cm^{-1} with 5400-7500 cm^{-1}.

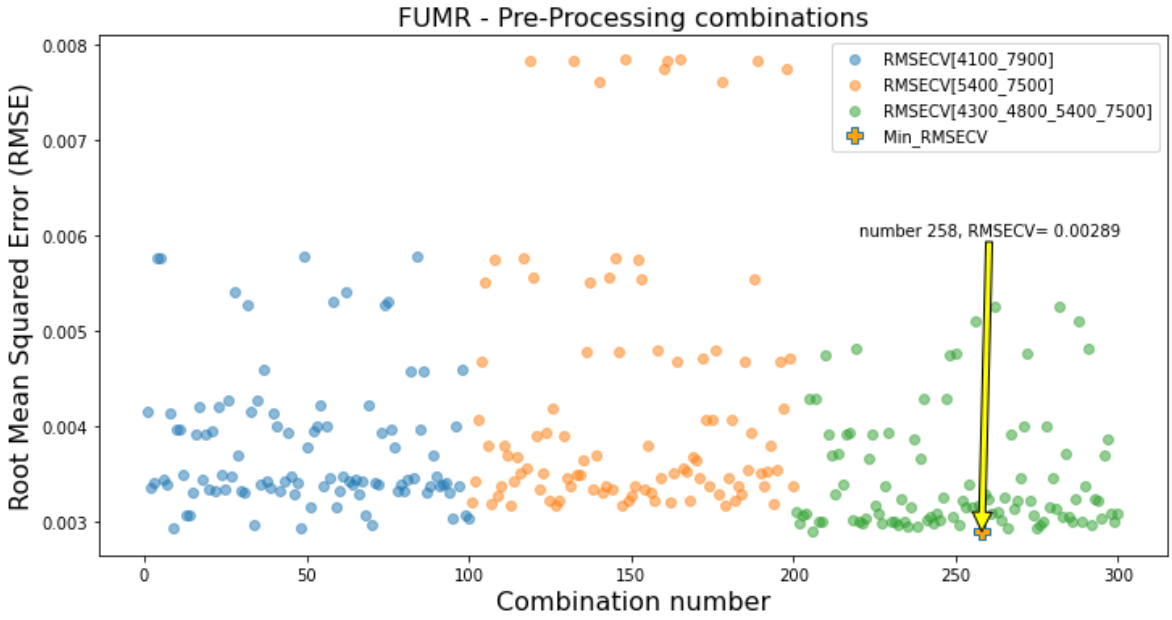


Figure 12 All PLS models generated to predict the FUMR in Python. Plot of the RMSECV vs the number of the specific preprocessing combination.

The best preprocessing combination is presented in the 258th position, corresponding to the following parameters and results (Table 6).

Table 6 Summary of the best PLS model developed in Python to predict FUMR.

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest	LV
Baseline + SG (deriv_order:1, filter_win: 13, poly_order: 3)	0.994	0.00225	0.990	0.00289	0.988	0.00286	15

The best preprocessing combines the baseline correction with Savitzky-Golay derivation with first-order derivative, 13 points of window filter, and third-order polynomial.

The FUMR model created in Python seems to be more representative of the resin since it uses the bands more related to the polymeric chain (5400-7500 cm^{-1}) and the region related to the bands of formaldehyde and urea (4300-4800 cm^{-1}). This model performs well like the one developed in OPUS with good RMSETest and R^2Test, as can be seen from the parity chart presented in Figure 13, where the closer to the diagonal line the results are, the more accurate the model is. Also, the prediction of the test set, in green, falls in the same position as the calibration set in blue.

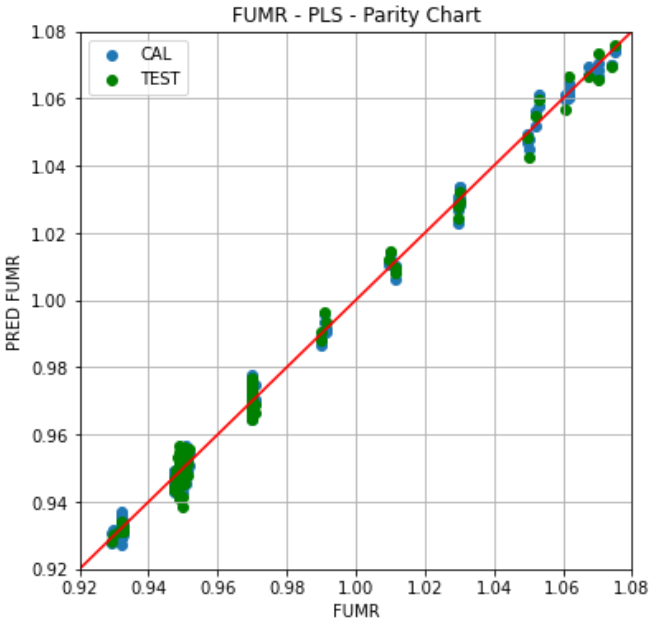


Figure 13 Parity chart for test validation of the best PLS model created in Python to predict FUMR.

4.3.2 Melamine content model - Python – PLS

The PLS melamine models were created using the same approach. As can be seen from Figure 14, the best results are presented in the same region of the spectra where the melamine bands typically appear (4300 - 4800 cm^{-1} and 5400 - 7500 cm^{-1}). However, the models did not perform well due to the melamine bands overlapped by the resin's principal components. Therefore, as the melamine models developed in OPUS, it is hard to find representative bands highly correlated to the resins' melamine content. The best approach is to use regions with bands related to the NH bonds presented in melamine, urea, methylolureas, methylolmelamines, and resin polymer.

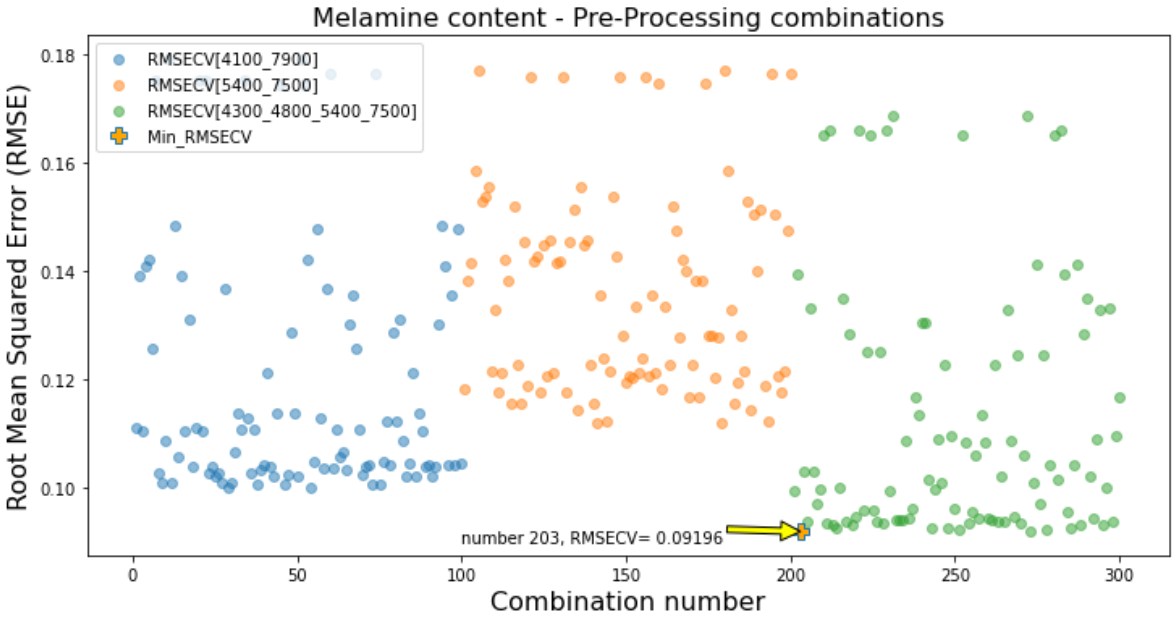


Figure 14 All PLS models generated to predict the melamine content in Python. Plot of the RMSECV vs the number of the specific preprocessing combination.

The best preprocessing for the melamine model is related to the 203rd position, which parameters and results are presented in Table 7.

Table 7 Summary of the best PLS model developed in Python to predict melamine content.

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest	LV
RNV IQR (90,10)	0.85016	0.07815	0.79252	0.09196	0.85249	0.09170	15

Unlike the FUMR model, the best melamine model uses only one preprocessing method. The selected pretreatment is the RNV with an interquartile range between the 90th and 10th percentile. The test validation has good performance compared to the calibration set, as can be seen in the parity chart presented in Figure 15. Despite the data's scattering, the test set's prediction falls in the same position as the calibration set, showing good predictability with prediction errors close to those presented in the reference method.

Since the resins' melamine content provided by the supplier has a maximum of up to 3.00 - 3.10 %, the model performance can be improved by eliminating the data above 3.25 % and through another study regarding more restricted bands.

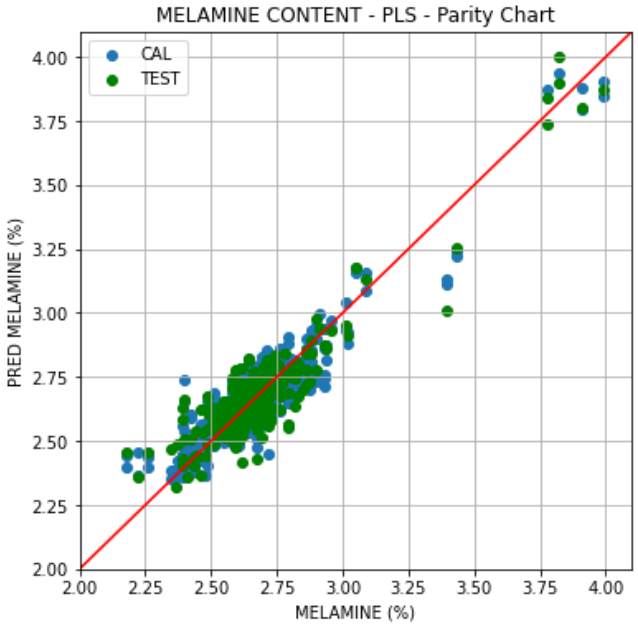


Figure 15 Parity chart for test validation of the best PLS model created in Python to predict melamine content.

4.3.3 Solids content model - Python - PLS

The solids content models have good performance as those created in OPUS, as can be seen from Figure 16. The best models are presented in the region more related to the polymeric chain (5400 - 7500 cm^{-1}), giving a good relationship between nonvolatile and volatile compounds at 120 ºC.

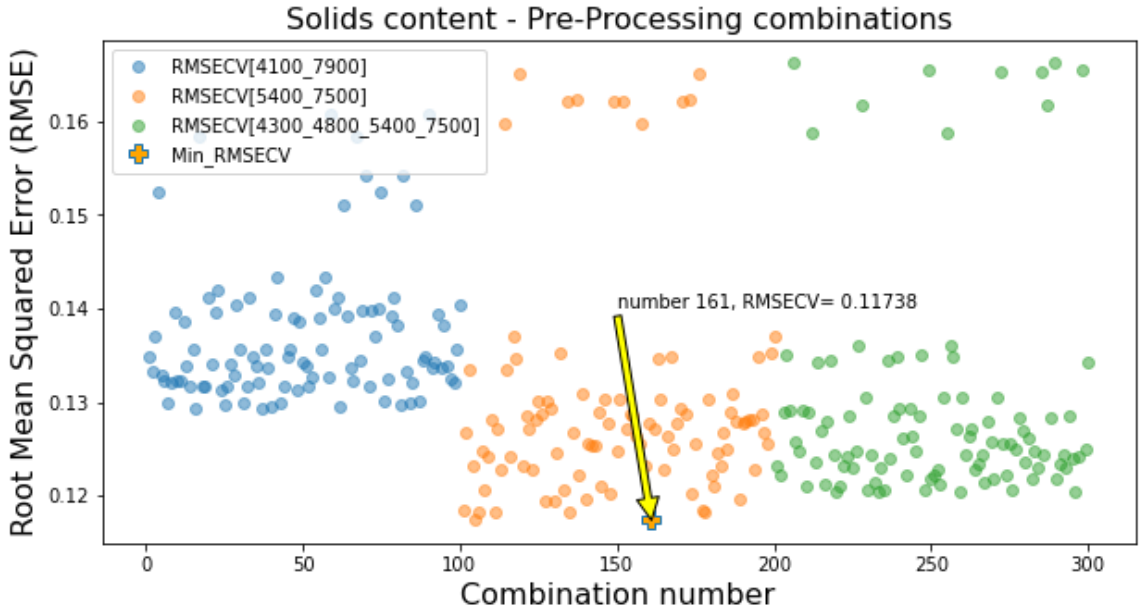


Figure 16 All PLS models generated to predict the solids content in Python. Plot of the RMSECV vs the number of the specific preprocessing combination.

The best preprocessing combination for the solids content model is presented in the 161st position, which parameters and results are shown in Table 8.

Table 8 Summary of the best PLS model developed in Python to predict solids content.

Preprocessing	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest	LV
Baseline + SG (deriv_order:1, filter_win: 7, poly_order: 3)	0.98604	0.09310	0.97782	0.11738	0.97866	0.11312	13

The best preprocessing is the combination of the baseline correction and SG derivation with first-order derivative, 7 points of window filter, and third-order polynomial. The SC model had a good performance as the FUMR model with R^2CV above 0.95. As expected, the test validation, which parity chart is presented in Figure 17, also had good values regarding the RMSETest and R^2Test.

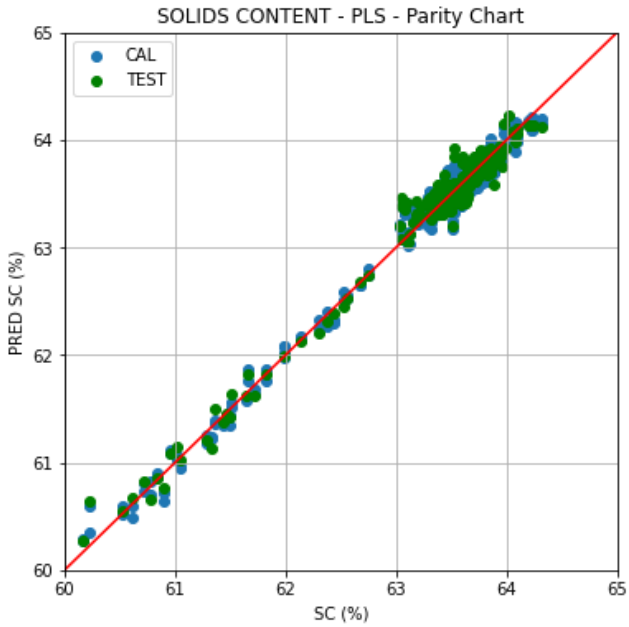


Figure 17 Parity chart for test validation of the best PLS model created in Python to predict solids content.

4.4 ANN models

The ANN models were created by applying PCA to the mean-centered and variance-scaled data of each spectral zone. As mentioned, the objective was to summarize the data and minimize computational time. After data reduction, the first approach was to select the best activation and optimization functions with a try-and-error procedure, analyzing the different functions

presented in the MLP regressor model from the scikit-learn library. The procedure was based on the analysis of the RMSECV and R^2CV for up to 30 neurons. As expected, the best performance was observed using the logistic sigmoid function, as described by Behera *et al*. and Alam *et al*., as the commonly used activation function in developing NIR models [38, 42]. For the optimization function, the best one was the 'lbfgs' due to its faster convergence and better performance, as described in the scikit-learn library [40].

The code shown in Annex D was used to create and analyze models' performance. The three main spectral regions were studied, and the results are presented next. The Analysis of the RMSECV and R^2CV related to the diverse numbers of neurons is shown in Annex E.

4.4.1 FUMR Model – ANN

The creation of the ANN model to predict FUMR started with the PCA of the data regarding each region of the resin's spectra. According to the PCA results, all regions had their data reduction made with four principal components, which explained variance corresponds to more than 99 %. In other words, 4 PCs can summarize more than 99 % of the data variation.

After, the number of neurons present in the hidden layer was studied. The number was selected based on the minimum RMSECV and maximum R^2CV; the results are presented in Table 9 and Annex E.

Table 9 Summary of the ANN models' results for the prediction of FUMR.

Spectral region (cm^{-1})	Number of neurons	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
4100-7900	30	0.99215	0.00263	0.96205	0.00579	0.95363	0.00559
4300-4800,5400-7500	23	0.99479	0.00214	0.98796	0.00326	0.98408	0.00328
5400-7500	30	0.95933	0.00599	0.836972	0.01200	0.85939	0.00975

According to Table 9, the best ANN model uses a hidden layer with 23 neurons and is related to the combination region of 4300 - 4800 with 5400 - 7500 cm^{-1}. This result confirms what was observed in the development of the PLS models. This region seems to be the best to predict this type of property due to the previously discussed representation bands. Its performance can be seen in the parity chart shown in Figure 18, where the test validation is visualized. With a R^2Test of more than 0.95, and RMSETest around the third decimal place, the parity chart represents the good predictability of the model.

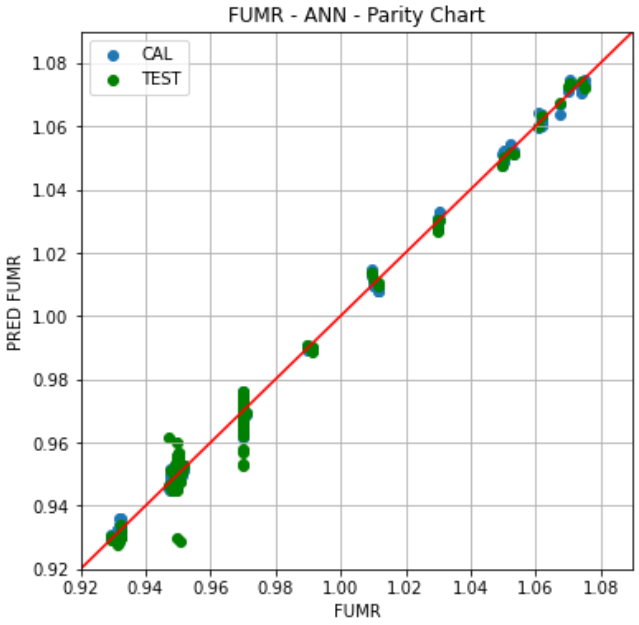


Figure 18 Parity chart for test validation of the best ANN model to predict FUMR.

4.4.2 Melamine content model – ANN

The melamine content model was developed using the same approach as the creation of the FUMR model. The data relating to each spectral zone were summarized with PCA, resulting in the same number of PCs (4). After reducing the data, the number of neurons in the hidden layer was studied. The results are presented in Table 10.

Table 10 Summary of the ANN models' results for the prediction of melamine content.

Spectral region (cm^{-1})	Number of neurons	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
4100-7900	5	0.38742	0.15802	0.26672	0.17289	0.31228	0.19801
4300-4800,5400-7500	5	0.52879	0.13859	0.34171	0.16381	0.50829	0.16743
5400-7500	14	0.70727	0.10923	0.29928	0.16900	0.486831	0.17105

As can be seen, the melamine content models have low performance in all regions, with R^2CV less than 0.50. Using five neurons in the hidden layer, the model with the highest R^2CV and lowest RMSECV is related to the combination region also used to develop the PLS models, as previously discussed. The parity chart in Figure 19 shows the model's behavior regarding test validation, where we visualize its low performance.

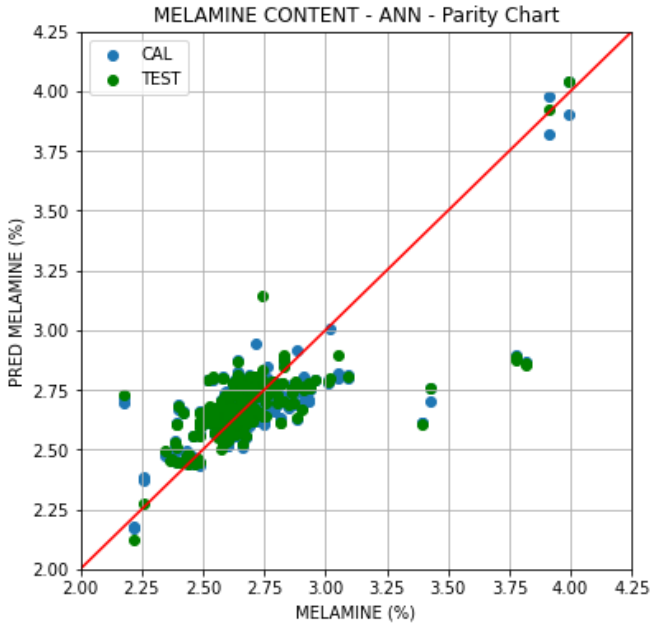


Figure 19 Parity chart for test validation of the best ANN model to predict melamine content.

In order to improve the model's performance, a preprocessing study can be done by adapting the code used to create the PLS models and eliminating irrelevant data. The code presented in annex D with the NIPPY tool makes it possible to analyze the best number of neurons for different data pretreatments. Also, the addition of another hidden layer can be an attempt to improve the performance, although it increases the model complexity.

4.4.3 Solids content model – ANN

Using the same criteria and as expected, the number of PCs used to reduce the solids content's data was the same as the others (around 4 with more than 99 % of explained variance). The solids content models' results are presented in Table 11.

Table 11 Summary of the ANN models' results for the prediction of solids content.

Spectral region (cm^{-1})	Number of neurons	R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
4100-7900	19	0.98538	0.09531	0.96705	0.14306	0.95907	0.15666
4300-4800,5400-7500	12	0.97364	0.12796	0.94168	0.19033	0.96977	0.13463
5400-7500	21	0.97530	0.12387	0.95362	0.16973	0.94195	0.18657

The models maintain the good performance related to the three studied spectral zones as observed in the PLS models. The best result uses 19 neurons in the hidden layer and is associated with all the important bands presented in the resin spectra, adapting well to the region related to the higher water absorbance (4800 and 5200 cm^{-1}).

The model's predictability regarding the validation test is visualized in the parity chart shown in Figure 20. The good accuracy and precision of the model can be seen since the prediction of the validation test falls on the diagonal line, almost in the same position as the calibration set.

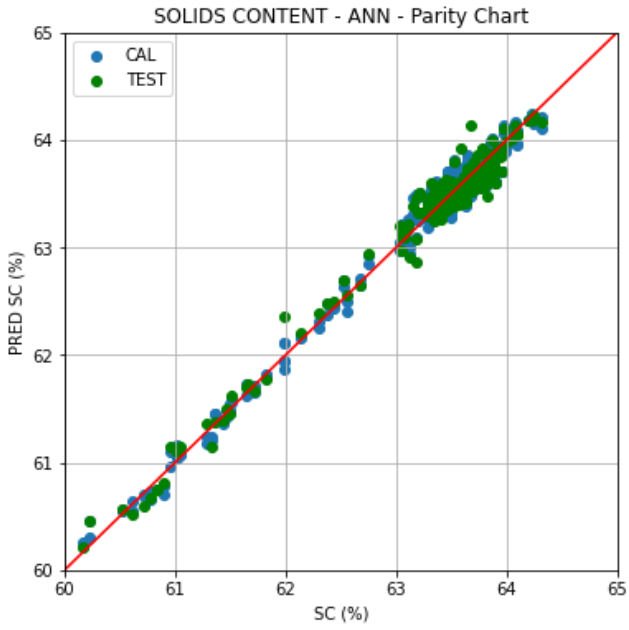


Figure 20 Parity chart for test validation of the best ANN model to predict solids content.

4.5 SVR models

Developing the SVR models was challenging because of the search process to select the proper data-dependent parameters. This process is usually as complicated and time-consuming as the one used to develop the ANN models. The code in Annex F shows the steps to develop these models, and its discussion is presented next.

After standardization of the spectra (prominent bands from 4100 - 7900 cm^{-1}) and properties, the best kernel function, C, and epsilon were selected through a try-and-error approach and by the minimum RMSECV and maximum R^2CV. The study started by analyzing two commonly used kernel functions ('linear' and 'RBF'). It was observed that the 'RBF', with the gamma parameter set to 'scale', had better performance than the 'linear' function regarding the computational time and accuracy for the same C and epsilon parameters (set 1 and 0.1, respectively, as default values presented in the scikit-learn library).

After selecting the best kernel function, the parameters C and epsilon were evaluated using different values. The parameter C was tested in the range from 1 to 1000. At the same time, the epsilon was tested, decreasing from 0.1 to 0.01 and 0.001.

Choosing a minimal epsilon and maximum C as possible to avoid overfitting, the SVR ignores the error posed by the data inside the margins ε and considers the rest for finding the optimal hyperplane with the help of the slack variables (ξ), as shown in Figure 7 [56]. During the

parameters study, the models had good performance for values of 1000 and 0.001 for C and epsilon, respectively.

The SVR models are presented next with the best results.

4.5.1 FUMR model – SVR

After selecting the best parameters using all the main bands, a study of the restricted spectral regions was performed, like the previous analysis, to determine the best FUMR model. The results are presented in Table 12.

Table 12 Summary of the SVR models' results for the prediction of FUMR.

Spectral region (cm^{-1})	Parameters		R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
	C	Epsilon						
4100-7900	1000	0.001	0.99999	0.00006	0.99490	0.00212	0.99323	0.00214
4300-4800,5400-7500	1000	0.001	0.99865	0.00109	0.99579	0.00193	0.99698	0.00143
5400-7500	1000	0.001	0.99241	0.00370	0.98450	0.00370	0.98793	0.00286

As can be seen, the models have good performance for the three studied spectral zones for the same parameters' values. The minimum RMSECV and maximum R^2CV were achieved in the combination region between 4300 - 4800 cm^{-1} and 5400 - 7500 cm^{-1}.

These results show how efficient the SVR is in developing regression models despite the difficulty in selecting the best parameters. Compared to the previous ones, the FUMR model developed with SVR has the best cross-validation and test validation performance. It can be seen from the parity chart shown in Figure 21 how accurate the model is regarding an independent data set.

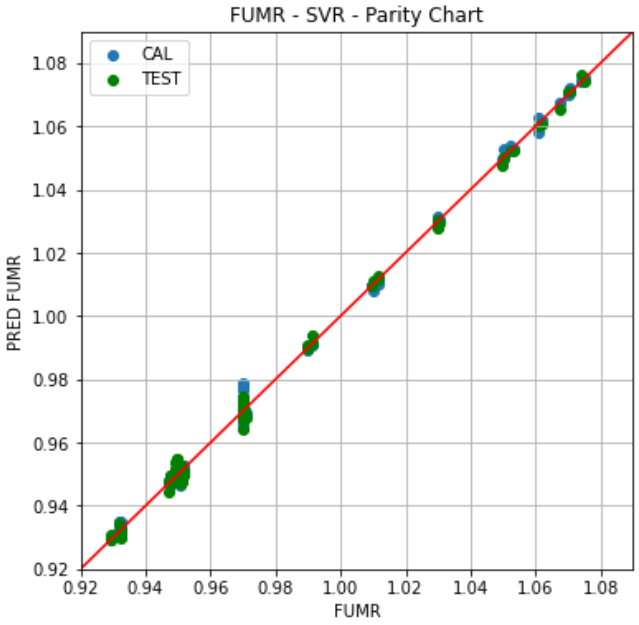


Figure 21 Parity chart for test validation of the best SVR model to predict FUMR.

4.5.2 Melamine content model – SVR

The development approach of the melamine model also started with selecting the best parameters, followed by studying the spectral zones. The results are presented in Table 13.

Table 13 Summary of the SVR models' results for the prediction of melamine content.

Spectral region (cm^{-1})	Parameters		R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
	C	Epsilon						
4100-7900	1000	0.001	0.91596	0.05853	0.63566	0.12187	0.79963	0.10688
4300-4800,5400-7500	1000	0.001	0.72979	0.10495	0.53569	0.13757	0.72895	0.12431
5400-7500	1000	0.001	0.44206	0.15081	0.29185	0.16990	0.33454	0.19478

The melamine models developed via SVR have a low performance like those created by ANN. The results show that the best model is associated with the spectral region related to the main resin's bands (4100 - 7900 cm^{-1}). Also, the notable difference between the statistical parameters of calibration and test validation shows the model's overfitting in which the model adjusts well to the calibration set only. The low efficiency can also be observed through the parity chart of the test validation presented in Figure 22. The scattering of the predicted results shows the low performance of the model regarding the test set.

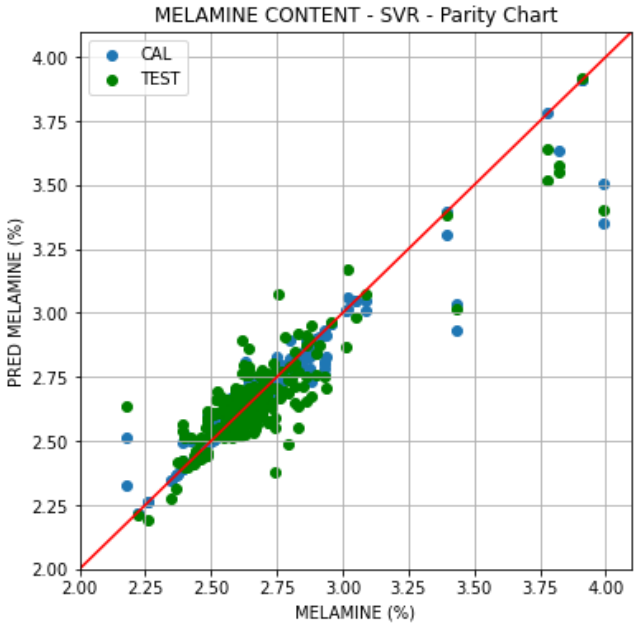


Figure 22 Parity chart for test validation of the best SVR model to predict melamine content.

In order to improve the model effectiveness, like was proposed in the ANN models, a preprocessing study can be done by adapting the PLS models' code and eliminating irrelevant data. The automatization can be done in order to relate the parameter C, epsilon, and preprocessing to the minimum RMSECV and maximum R^2CV.

4.5.3 Solids content model – SVR

The solids content models perform similarly to the FUMR models, with good cross-validation and test validation results. It can be seen from Table 14 that the models have good performance regarding the three spectral regions as well.

Table 14 Summary of the SVR models' results for the prediction of solids content.

Spectral region (cm^{-1})	Parameters		R^2Cal	RMSECal	R^2CV	RMSECV	R^2Test	RMSETest
	C	Epsilon						
4100-7900	1000	0.001	0.98161	0.03379	0.97943	0.11304	0.98006	0.10935
4300-4800,5400-7500	1000	0.001	0.99215	0.06983	0.98164	0.10677	0.98772	0.08583
5400-7500	1000	0.001	0.98317	0.10224	0.97654	0.12070	0.97817	0.11440

According to Table 14, the model with the best result is related to the combination of the two regions, 4300 - 4800 cm^{-1} and 5400 - 7500 cm^{-1}. Also, like the FUMR model developed with SVR, it shows the best performance regarding all the SC models created in this work. The model's effectiveness can be seen in Figure 23 regarding the test set validation.

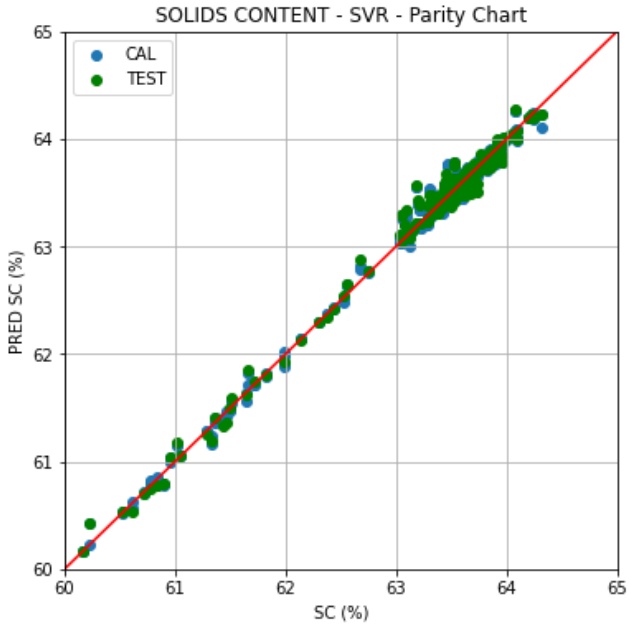


Figure 23 Parity chart for test validation of the best SVR model to predict solids content

The model is very accurate, as seen by the parity chart's diagonal line. There is a minor difference between the predicted and reference values showing the good fitting for both calibration and test sets.

5 Conclusions

The models created in both platforms, OPUS and Python, have a good performance as observed through cross-validation and test validation results, mainly regarding FUMR and SC models. The melamine content models need more attention in order to enhance their predictive ability through data manipulation like using more restricted bands or removing irrelevant information.

The selection of the spectral regions was crucial to the development of the models. The results show that all three studied regions seemed to be the best resin representative bands. Although, the selection of the more distinguished ones showed better performance, mainly with the elimination of the water band between 4800 and 5200 cm^{-1}. In some cases, like in SC-ANN and melamine content-SVR models, all the main bands of the resin's spectrum (4100 - 7900 cm-1) were used, adapting to the typical noisy band related to water. It occurs because the ANN and SVR are versatile methodologies. In other words, the model can adjust well to linear or non-linear correlated data by selecting the correct parameters.

The creation of the models in OPUS was essential due to its use by the company. Since the operators already know how to handle it, the models can be easily applied after transferring the files to the computer where the spectra acquisitions are made. Considering OPUS is a closed-source software, it has some limitations if compared to Python; it does not have various manipulations like other types of regression, featuring selection or preprocessing combinations, for example.

Except for the melamine content, all regression models had practically the same performance on both platforms. Although, by a slight difference, the best ones were selected. The models developed in Python had the best performance using PLS for melamine content and SVR for FUMR and SC. The best FUMR and SC models had values more than 0.95 for R^2CV and R^2Test, and minimal RMSECV and RMSETest. The best melamine model had R^2CV and R^2Test of 0.79 and 0.85, respectively, and RMSECV and RMSETest of around 0.1. The PLS models created in OPUS also had similar efficiency and can be promptly applied by the company.

The more significant difference is related to the melamine models; the ones developed with ANN and SVR did not perform well and, as mentioned before, can be manipulated to improve their performance. In comparison, despite the absent bands more related to melamine, the ones developed with PLS worked better due to preprocessing with predictive errors close to those presented in the reference method.

6 Assessment of the work done

6.1 Objectives Achieved

The main goals of this dissertation were achieved entirely. Three different regression types were used to develop the NIR models (PLS, ANN, and SVR). The models created in both platforms, OPUS and Python, have good predictive ability showing good results regarding the validation processes. Since the application of Python's models needs a certain programming knowledge, which consequently becomes more challenging to apply, the models developed in OPUS are available and can be easily implemented by the company.

Even though the results obtained on both platforms are not much different, the best Python results show the versatility of an open-source platform compared to a commercial software.

6.2 Other Work Carried Out

During this work, OPUS models were also developed using the entire database to have more information within the calibration set. These models were tested, through cross-validation and external validation, and performed similarly to those presented in this dissertation. Also, a solids content model was created with the same approach to analyze resins produced in Portugal with the same characteristics as those used in Germany.

OPUS models to predict resin's pH and viscosity were also made but were discarded due to low performance.

6.3 Final Assessment

The development of this dissertation allowed me to improve my knowledge regarding NIR spectroscopy, amino resins, and regression methods. Also, it permitted me to learn how the main resins' properties are measured and improve my experience in the laboratory. In addition, the use of Python programming allowed me to get experience in an important and vast subject called machine learning, an essential subfield of artificial intelligence widely used for prediction across many fields.

In conclusion, I hope that the summarized scientific information and the models' development approach presented in this dissertation can also help the company create new predictive models for the characterization of similar resins.

7 References

[1] S. Arauco, "Sonae Arauco Taking Wood Further," 2022. https://www.sonaearauco.com/en/company/who-we-are_2455.html (accessed May 01, 2022).

[2] A. Pizzi and K. L. Mittal, *"Chapter 10 - Urea and Melamine Amino resin Adhesives," in Handbook of Adhesive Technology, Third Edition*. CRC Press, 2017. doi: 10.1201/9781315120942.

[3] IUPAC, *The IUPAC Compendium of Chemical Terminology*. Research Triangle Park, NC: International Union of Pure and Applied Chemistry (IUPAC), 2019. doi: 10.1351/goldbook.

[4] S. Tohmura, A. Inoue, and S. H. Sahari, "Influence of the melamine content in melamine-urea-formaldehyde resins on formaldehyde emission and cured resin structure," *Journal of Wood Science*, vol. 47, no. 6, pp. 451–457, 2001, doi: 10.1007/BF00767897.

[5] M. Dunky, "Urea–formaldehyde (UF) adhesive resins for wood," *International Journal of Adhesion and Adhesives*, vol. 18, no. 2, pp. 95–107, 1998, doi: https://doi.org/10.1016/S0143-7496(97)00054-7.

[6] A. Henriques, P. Cruz, J. M. Ferra, J. Martins, F. D. Magalhães, and L. H. Carvalho, "Determination of melamine content in amino resins by near-infrared spectroscopy," *Wood Science and Technology*, vol. 47, no. 5, pp. 939–948, 2013, doi: 10.1007/s00226-013-0547-6.

[7] M. Gonçalves, N. T. Paiva, J. M. Ferra, J. Martins, F. Magalhães, and L. Carvalho, "Near infrared spectroscopy for rapid determination of solids content of amino resins," *Journal of Near Infrared Spectroscopy*, vol. 28, no. 5–6, pp. 344–350, Aug. 2020, doi: 10.1177/0967033520947824.

[8] I. Whiteside, "Process for the preparation of urea-formaldehyde resins," US4968773A, Nov. 06, 1990

[9] H. W. Siesler, *"Chapter 2 - Basic Principles of Near-Infrared Spectroscopy," in Handbook of near-infrared analysis, Third Edition*. Boca Raton: CRC Press, 2008. [Online]. Available: https://doi.org/10.1201/9781420007374

[10] M. Blanco and I. Villarroya, "NIR spectroscopy: a rapid-response analytical tool," *TrAC Trends in Analytical Chemistry*, vol. 21, no. 4, pp. 240–250, 2002, doi: https://doi.org/10.1016/S0165-9936(02)00404-1.

[11] Jr. , J. Workman and L. Weyer, *"Chapter 1 - Introduction to Near-Infrared Spectra," in Practical Guide to Interpretive Near-Infrared Spectroscopy, First Edition*. CRC Press, 2007. doi: 10.1201/9781420018318.

[12] M. Gonçalves, "Implementation of a near infrared methodology for quality control of amino resins," Doctoral program in Refining, Petrochemical and Chemical Engineering, Faculty of engineering of the University of Porto, 2020.

[13] J. Workman and D. Burns, *"Chapter 4 - Commercial NIR Instrumentation," in Handbook of Near-Infrared Analysis*. CRC Press, 2007. doi: 10.1201/9781420007374.

[14] J. Skvaril, K. G. Kyprianidis, and E. Dahlquist, "Applications of near-infrared spectroscopy (NIRS) in biomass energy conversion processes: A review," *Applied Spectroscopy Reviews*, vol. 52, no. 8, pp. 675–728, Sep. 2017, doi: 10.1080/05704928.2017.1289471.

[15] B. K. Lavine, "Chemometrics," *Analytical Chemistry*, vol. 70, no. 12, pp. 209–228, Jun. 1998, doi: 10.1021/a19800085.

[16] H. W. Siesler, "Vibrational Spectroscopy," in *Reference Module in Materials Science and Materials Engineering*, Elsevier, 2016. doi: https://doi.org/10.1016/B978-0-12-803581-8.01318-7.

[17] P. Geladi and E. Dåbakk, "An Overview of Chemometrics Applications in near Infrared Spectrometry," *Journal of Near Infrared Spectroscopy*, vol. 3, no. 3, pp. 119–132, Jun. 1995, doi: 10.1255/jnirs.63.

[18] B. K. Lavine, "Chemometrics," *Analytical Chemistry*, vol. 70, no. 12, pp. 209–228, Jun. 1998, doi: 10.1021/a19800085.

[19] A. Henriques, P. Cruz, J. Martins, J. M. Ferra, F. D. Magalhães, and L. H. Carvalho, "Determination of formaldehyde/urea molar ratio in amino resins by near-infrared spectroscopy," *Journal of Applied Polymer Science*, vol. 124, no. 3, pp. 2441–2448, May 2012, doi: https://doi.org/10.1002/app.35128.

[20] M. Gonçalves, N. T. Paiva, J. M. Ferra, J. Martins, F. D. Magalhães, and L. Carvalho, "Chemical composition of melamine-urea-formaldehyde (MUF) resins assessed by near-infrared (NIR) spectroscopy," *International Journal of Adhesion and Adhesives*, vol. 93, p. 102327, 2019, doi: https://doi.org/10.1016/j.ijadhadh.2019.01.021.

[21] H. Kasprzyk, M. Jozwiak, and S. Proszyk, "Application of NIR spectroscopy for analysis of amino adhesive resins applied in wood based materials.," *Folia Forestalia Polonica*, vol. 32, pp. 67–74, 2001.

[22] M. Boysworth and K. Booksh, *"Chapter 10 - Aspects of Multivariate Calibration Applied to Near-Infrared Spectroscopy, " in Handbook of Near-Infrared Analysis*. CRC Press, 2007. doi: 10.1201/9781420007374.

[23] Å. Rinnan, F. van den Berg, and S. B. Engelsen, "Review of the most common pre-processing techniques for near-infrared spectra," *TrAC Trends in Analytical Chemistry*, vol. 28, no. 10, pp. 1201–1222, 2009, doi: https://doi.org/10.1016/j.trac.2009.07.007.

[24] Bruker Optik, "Reference Manual - Opus Spectroscopy software." p. 120, 2004.

[25] Q. Guo, W. Wu, and D. L. Massart, "The robust normal variate transform for pattern recognition with near-infrared data," *Analytica Chimica Acta*, vol. 382, no. 1, pp. 87–103, 1999, doi: https://doi.org/10.1016/S0003-2670(98)00737-5.

[26] M. Zeaiter and D. Rutledge, "3.04 - Preprocessing Methods," in *Comprehensive Chemometrics*, S. D. Brown, R. Tauler, and B. Walczak, Eds. Oxford: Elsevier, 2009, pp. 121–231. doi: https://doi.org/10.1016/B978-044452701-1.00074-0.

[27] Y. Roggo, P. Chalus, L. Maurer, C. Lema-Martinez, A. Edmond, and N. Jent, "A review of near infrared spectroscopy and chemometrics in pharmaceutical technologies," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 44, no. 3, pp. 683–700, 2007, doi: https://doi.org/10.1016/j.jpba.2007.03.023.

[28] R. M. Balabin, R. Z. Safieva, and E. I. Lomakina, "Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 88, no. 2, pp. 183–188, 2007, doi: https://doi.org/10.1016/j.chemolab.2007.04.006.

[29] B. Lavine, "A user-friendly guide to multivariate calibration and classification, Tomas Naes, Tomas Isakson, Tom Fearn and Tony Davies, NIR Publications, Chichester, 2002, ISBN 0-9528666-2-5.," *Journal of Chemometrics*, vol. 17, no. 10, pp. 571–572, Oct. 2003, doi: https://doi.org/10.1002/cem.815.

[30] D. Ballabio, "A MATLAB toolbox for Principal Component Analysis and unsupervised exploration of data structure," *Chemometrics and Intelligent Laboratory Systems*, vol. 149, pp. 1–9, 2015, doi: https://doi.org/10.1016/j.chemolab.2015.10.003.

[31] B. Park, Y. R. Chen, W. R. Hruschka, S. D. Shackelford, and M. Koohmaraie, "Principal component regression of near-infrared reflectance spectra for beef tenderness prediction," *Transactions of the ASAE*, vol. 44, no. 3, p. 609, 2001, doi: https://doi.org/10.13031/2013.6087.

[32] W. J. Dunn, D. R. Scott, and W. G. Glen, “Principal components analysis and partial least squares regression,” *Tetrahedron Computer Methodology*, vol. 2, no. 6, pp. 349–376, 1989, doi: https://doi.org/10.1016/0898-5529(89)90004-3.

[33] A. L. Leal, A. M. S. Silva, J. C. Ribeiro, and F. G. Martins, “Data driven models exploring the combination of NIR and 1H NMR spectroscopies in the determination of gasoline properties,” *Microchemical Journal*, vol. 175, p. 107217, 2022, doi: https://doi.org/10.1016/j.microc.2022.107217.

[34] L. Nørgaard, A. Saudland, J. Wagner, J. P. Nielsen, L. Munck, and S. B. Engelsen, “Interval Partial Least-Squares Regression (iPLS): A Comparative Chemometric Study with an Example from Near-Infrared Spectroscopy,” *Applied Spectroscopy*, vol. 54, no. 3, pp. 413–419, Mar. 2000, doi: 10.1366/0003702001949500.

[35] Z. Ran, L. Sun, Y. Liu, X. Pan, J. Li, and Y. Liu, “Forward and backward interval partial least squares method for quantitative analysis of frying oil quality,” *Infrared Physics & Technology*, vol. 105, p. 103207, 2020, doi: https://doi.org/10.1016/j.infrared.2020.103207.

[36] X. Zuo, S. Fang, and X. Liang, “Synergy Interval Partial Least Square (siPLS) with Potentiometric Titration Multivariate Calibration for the Simultaneous Determination of Amino Acids in Mixtures,” *Advance Journal of Food Science and Technology*, vol. 6, no. 11, pp. 1209–1218, Nov. 2014, doi: 10.19026/ajfst.6.187.

[37] İ. Akyüz, Ş. Özşahin, S. Tiryaki, and A. Aydın, “An application of artificial neural networks for modeling formaldehyde emission based on process parameters in particleboard manufacturing process,” *Clean Technologies and Environmental Policy*, vol. 19, no. 5, pp. 1449–1458, 2017, doi: 10.1007/s10098-017-1342-0.

[38] S. K. Behera, S. K. Meher, and H.-S. Park, “Artificial neural network model for predicting methane percentage in biogas recovered from a landfill upon injection of liquid organic waste,” *Clean Technologies and Environmental Policy*, vol. 17, no. 2, pp. 443–453, 2015, doi: 10.1007/s10098-014-0798-4.

[39] D. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *International Conference on Learning Representations*, Dec. 2014.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-Learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[41] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. Vanderplas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," Sep. 2013.

[42] K. Alam, S. Stanton, and G. Hebner, *"Chapter 36 - Part A - Resin Identification Using Near-Infrared Spectroscopy and Neural Networks," in Handbook of Near-Infrared Analysis*. 2007.

[43] R. Ghasemiyeh, R. Moghdani, and S. S. Sana, "A Hybrid Artificial Neural Network with Metaheuristic Algorithms for Predicting Stock Price," *Cybernetics and Systems*, vol. 48, no. 4, pp. 365–392, May 2017, doi: 10.1080/01969722.2017.1285162.

[44] D. Pérez-Marín, A. Garrido-Varo, and J. E. Guerrero, "Non-linear regression methods in NIRS quantitative analysis," *Talanta*, vol. 72, no. 1, pp. 28–42, 2007, doi: https://doi.org/10.1016/j.talanta.2006.10.036.

[45] I. Kaastra and M. Boyd, "Designing a neural network for forecasting financial and economic time series," *Neurocomputing*, vol. 10, no. 3, pp. 215–236, 1996, doi: https://doi.org/10.1016/0925-2312(95)00039-9.

[46] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995, doi: 10.1007/BF00994018.

[47] R. M. Balabin and E. I. Lomakina, "Support vector machine regression (SVR/LS-SVM)—an alternative to neural networks (ANN) for analytical chemistry? Comparison of nonlinear methods on near infrared (NIR) spectroscopy data," *Analyst*, vol. 136, no. 8, p. 1703, 2011, doi: 10.1039/c0an00387e.

[48] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004, doi: 10.1023/B:STCO.0000035301.49549.88.

[49] K. Cheng, Z. Lu, Y. Wei, Y. Shi, and Y. Zhou, "Mixed kernel function support vector regression for global sensitivity analysis," *Mechanical Systems and Signal Processing*, vol. 96, pp. 201–214, 2017, doi: https://doi.org/10.1016/j.ymssp.2017.04.014.

[50] J. Torniainen, I. O. Afara, M. Prakash, J. K. Sarin, L. Stenroth, and J. Töyräs, "Open-source python module for automated preprocessing of near infrared spectroscopic data," *Analytica Chimica Acta*, vol. 1108, pp. 1–9, 2020, doi: https://doi.org/10.1016/j.aca.2020.02.030.

[51] A. C. Olivieri, *Introduction to Multivariate Calibration*. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-97097-4.

[52] N. Costa, S. Amaral, R. Alvim, M. Nogueira, M. Schwanninger, and J. Rodrigues, "Assessment of resin formulations and determination of the formaldehyde to urea molar ratio by near- and mid-infrared spectroscopy and multivariate data analysis," *Journal of Applied Polymer Science*, vol. 128, no. 1, pp. 498–508, Apr. 2013, doi: https://doi.org/10.1002/app.38206.

[53] Jr. , J. Workman and L. Weyer, *"Appendix 4a - Spectra-Structure Correlations for Near Infrared," in Practical Guide to Interpretive Near-Infrared Spectroscopy, First Edition*. CRC Press, 2007. doi: 10.1201/9781420018318.

[54] X. Lian, M. Zhang, X. Sun, C. Song, H. Yuan, L. Guo, and X. Li, "Online real time determination of free formaldehyde content during polymerization process of phenolic resin by NIR spectra and a modeling-free method," *Polymer Testing*, vol. 93, p. 106584, 2021, doi: https://doi.org/10.1016/j.polymertesting.2020.106584.

[55] L. J. Mauer, A. A. Chernyshova, A. Hiatt, A. Deering, and R. Davis, "Melamine Detection in Infant Formula Powder Using Near- and Mid-Infrared Spectroscopy," *Journal of Agricultural and Food Chemistry*, vol. 57, no. 10, pp. 3974–3980, May 2009, doi: 10.1021/jf900587m.

[56] R. Gholami and N. Fakhari, "Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications," in *Handbook of Neural Computation*, P. Samui, S. Sekhar, and V. E. Balas, Eds. Academic Press, 2017, pp. 515–535. doi: https://doi.org/10.1016/B978-0-12-811318-9.00027-2.

Annex A - Complementary results for the models created in OPUS

Table 15 Summary of the OPUS - PLS models for FUMR using the wavenumber range 4100-7900 cm^{-1}

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.9876	0.00336	0.9828	0.0039	
Min-Max norm.	0.9882	0.00328	0.9832	0.00385	
MSC	0.9883	0.00328	0.9836	0.00381	15
SNV	0.9883	0.00327	0.9836	0.00381	
COE	0.9877	0.00335	0.9826	0.00392	

Table 16 Summary of the OPUS - PLS models for FUMR using the wavenumber ranges 4300-4800 and 5400-7500 cm^{-1}

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.9924	0.00263	0.9887	0.00316	
SNV	0.9936	0.00242	0.9898	0.00300	
Min-Max Norm.	0.9932	0.00250	0.9896	0.00303	15
COE	0.9927	0.00259	0.9889	0.00313	
1^{st} derivative + SNV	0.9935	0.00243	0.9895	0.00305	

Table 17 Summary of the OPUS - PLS models for Melamine content using the wavenumber range 4100-7900 cm^{-1}

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.7620	0.1000	0.6688	0.116	
SLS	0.7783	0.0967	0.6833	0.114	
COE	0.7797	0.0964	0.6879	0.113	15
MSC	0.7697	0.0985	0.6721	0.116	
SNV	0.7722	0.09800	0.6720	0.116	

Table 18 Summary of the OPUS - PLS models for Melamine content using the wavenumber range 5400-7500 cm⁻

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.7776	0.0968	0.6562	0.118	
MSC	0.8091	0.0897	0.6747	0.115	
Min-Max Norm.	0.8044	0.0908	0.6739	0.115	15
SNV	0.7999	0.0918	0.6714	0.116	
SLS	0.8124	0.0889	0.6695	0.116	

Table 19 Summary of the OPUS - PLS models for Solids content using the wavenumber range 4100-7900 cm^{-1}

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.9747	0.128	0.9641	0.149	
SNV	0.9759	0.124	0.9648	0.148	
MSC	0.9757	0.125	0.9647	0.148	15
COE	0.9747	0.128	0.9634	0.151	
SLS	0.9745	0.128	0.9636	0.150	

Table 20 Summary of the OPUS - PLS models for Solids content using the wavenumber ranges 4300-4800 and 5400-7500 cm^{-1}

Preprocessing	R2Cal	RMSECal	R2CV	RMSECV	LV
-	0.9831	0.104	0.9749	0.125	
Min-Max Norm.	0.9832	0.104	0.9751	0.124	
COE	0.9831	0.104	0.9750	0.125	15
1^{st} derivative + SNV	0.9850	0.0983	0.9749	0.125	
SNV	0.9835	0.103	0.9750	0.124	

Annex B - PLS code

```
# Loading Python packages

from sys import stdout
from openpyxl import Workbook, load_workbook
from sklearn.preprocessing import StandardScaler
from sklearn.cross_decomposition import PLSRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import LeaveOneOut
import os
import nippy
import linecache
import pandas as pd
import numpy as np
```

```
# Selection of the calibration set presented in excel files
# df_input= spectra + wavelengths, df_out= all properties

df_input=pd.read_excel('11G300 INPUT CAL.xlsx',header=None)
df_out=pd.read_excel('11G300 OUTPUT CAL.xlsx')

# Selection of the desired property

df_prop=df_out.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_cal=df_input.loc[0,:]
df_spect_cal=df_input.loc[1:,:]
```

```
# Selection of the test set presented in excel files
# df_input_test= spectra + wavelengths, df_out_test= all properties

df_input_test=pd.read_excel('11G300 INPUT TEST.xlsx',header=None)
df_out_test=pd.read_excel('11G300 OUTPUT TEST.xlsx')

# Selection of the desired property

df_prop_test=df_out_test.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_test=df_input_test.loc[0,:]
df_spect_test=df_input_test.loc[1:,:]
```

```
# NIPPY - Spectra Preprocessing
    # 1. Load configuration presented in a '.ini' file extension

pipelines = nippy.read_configuration('MELAMINE_PLS_PIPELINE.ini')

    # 2. Load data - conversion to numpy

wavelength_cal = df_wave_cal.T.to_numpy() #Rows = wavelength, Columns = samples
wavelength_test = df_wave_test.T.to_numpy()
spectra_calibration = df_spect_cal.T.to_numpy()
spectra_test = df_spect_test.T.to_numpy()

    # 3. Dataset through all pipelines

datasets_calibration = nippy.nippy(wavelength_cal, spectra_calibration, pipelines)
datasets_test = nippy.nippy(wavelength_test, spectra_test, pipelines)

    # 4. Export the preprocessed data

nippy.export_pipelines_to_csv('PRE_PRO_MEL_PLS_CAL', datasets_calibration, pipelines, \
mkdir=True)
nippy.export_pipelines_to_csv('PRE_PRO_MEL_PLS_TEST', datasets_test, pipelines, mkdir=True)
```

```
# Function to automate PLS models' creation.
# cal_path - Location of the preprocessed spectra of the calibration set - csv files
# test_path - Location of the preprocessed spectra of the test set - csv files
# cal_pipelines - Location of the preprocessing type performed in calibration set - txt file
# test_pipelines - Location of the preprocessing type performed in test set - txt file
# lv_number - number of latent variables to analyze
# excel_wb - Excel file to save the results
# excel_ws - Excel sheet where the results are saved

def pls_spec_analysis(cal_path,test_path,cal_pipelines,test_pipelines, \
    lv_number,excel_wb,excel_ws):
    # Number of pipelines created - equal to calibration and test sets

    with open(cal_pipeline, 'r') as fp:
        for count, line in enumerate(fp):
            pass
        print("Number of pipelines:",count+1)
        line_numbers = range(1,count+2)

        # Walk through the preprocessed calibration and test sets directories

        for (root1, dirs1, files1),(root2,dirs2,files2) in zip(
             os.walk(cal_path),os.walk(test_path)):

            # Selection of the preprocessed data of the calibration and test sets

            for i,j,k in zip(sorted(files1,key=len),sorted(files2,key=len),line_numbers):
```

```
# Print the preprocessing type performed in the calibration set

line1 = linecache.getline(cal_pipelines, k).strip()
print('CAL_PIPELINE:',line1)

# Calibration set after preprocessing

data1 = np.genfromtxt(os.path.join(root1, i), delimiter=',')
spec_cal = data1[:,1:]  # Rows = wavelength, Columns = samples
wave1 = data1[:,0]
print(i) # print file name

# Print the preprocessing type performed in the test set

line2 = linecache.getline(test_pipelines, k).strip()
print('TEST_PIPELINE:',line2)

# Test set after preprocessing

data2 = np.genfromtxt(os.path.join(root2, j), delimiter=',')
spec_test = data2[:,1:]  # Rows = wavelength, Columns = samples
wave2 = data2[:,0]
print(j) # print file name

# Definition of x and y variables

X_cal=spec_cal.T
X_test=spec_test.T
y_cal=df_prop
y_test=df_prop_test

# Normalization of the x variables

scx= StandardScaler()
scx.fit(spec_cal.T)
X_calN=scx.transform(X_cal)
X_testN=scx.transform(X_test)

# Normalization of the y variables

scy= StandardScaler()
scy.fit(y_cal)
y_calN=scy.transform(y_cal)
y_testN=scy.transform(y_test)
```

```
# root mean squared error

RMSECV=[]
RMSEcal=[]
RMSEtest=[]

# Coefficient of determination

R2CV=[]
R2cal=[]
R2test=[]

# Analysis of the PLS model

xi=np.arange(1,lv_number+1,1) # Total number of latent variables to analyze
for j in xi:
    PLS=PLSRegression(n_components=j)

    # Fit model to data

    PLS.fit(X_calN,y_calN)

    # Leaving-one-out cross-validation prediction

    loo = LeaveOneOut()
    y_CV = cross_val_predict(PLS, X_calN, y_calN, cv=loo,n_jobs=-1)

    # Prediction of the properties by the model

    y_calP=PLS.predict(X_calN)
    y_testP=PLS.predict(X_testN)

    # Scale back the data to the original representation

    y_CV=scy.inverse_transform(y_CV)
    y_calP=scy.inverse_transform(y_calP)
    y_testP=scy.inverse_transform(y_testP)

    # Calculate the root mean squared errors

    rmsecal=mean_squared_error(y_cal,y_calP,squared=False)
    rmseCV=mean_squared_error(y_cal, y_CV,squared= False)
    rmsetest=mean_squared_error(y_test,y_testP,squared=False)
    RMSEcal.append(rmsecal)
    RMSECV.append(rmseCV)
    RMSEtest.append(rmsetest)
```

```
                    # Calculate the scores

                    r2cal=r2_score(y_cal,y_calP)
                    r2CV=r2_score(y_cal, y_CV)
                    r2test=r2_score(y_test,y_testP)
                    R2cal.append(r2cal)
                    R2CV.append(r2CV)
                    R2test.append(r2test)

                    # Update status on the same line

                    comp = 100*(j+1)/(lv_number+1)
                    stdout.write("\r%d%% completed" % comp)
                stdout.flush()
                stdout.write("\n")

                #position of the maximum R2CV

                r2maxCV = np.argmax(R2CV)

                #position of the minimum RMSECV

                rmseminCV = np.argmin(RMSECV)

                # Access sheet to save the results

                wb = load_workbook(excel_wb)
                ws = wb[excel_ws]
                to_append = (
                [line1,
                RMSEcal[rmseminCV],
                RMSECV[rmseminCV],
                RMSEtest[rmseminCV],
                R2cal[r2maxCV],
                R2CV[r2maxCV],
                R2test[r2maxCV],
                rmseminCV+1]
                )
                ws.append(to_append)
                wb.save(excel_wb)
```

```
# Example of how to apply the function

cal_path= 'C:\\Users\\selen\\PRE_PRO_MEL_PLS_CAL'
test_path= 'C:\\Users\\selen\\PRE_PRO_MEL_PLS_TEST'
cal_pipeline="C:\\Users\\selen\\PRE_PRO_MEL_PLS_CAL\\pipelines.log"
test_pipeline="C:\\Users\\selen\\PRE_PRO_MEL_PLS_TEST\\pipelines.log"
lv_number=15
excel_file= "C:\\Users\\selen\\PYPLS_test.xlsx"
excel_sheet= 'melamine_pls'
pls_spec_analysis(cal_path,test_path,cal_pipeline,test_pipeline,lv_number, \
                  excel_file,excel_sheet)
```

Annex C - Complementary results for the PLS models created in Python

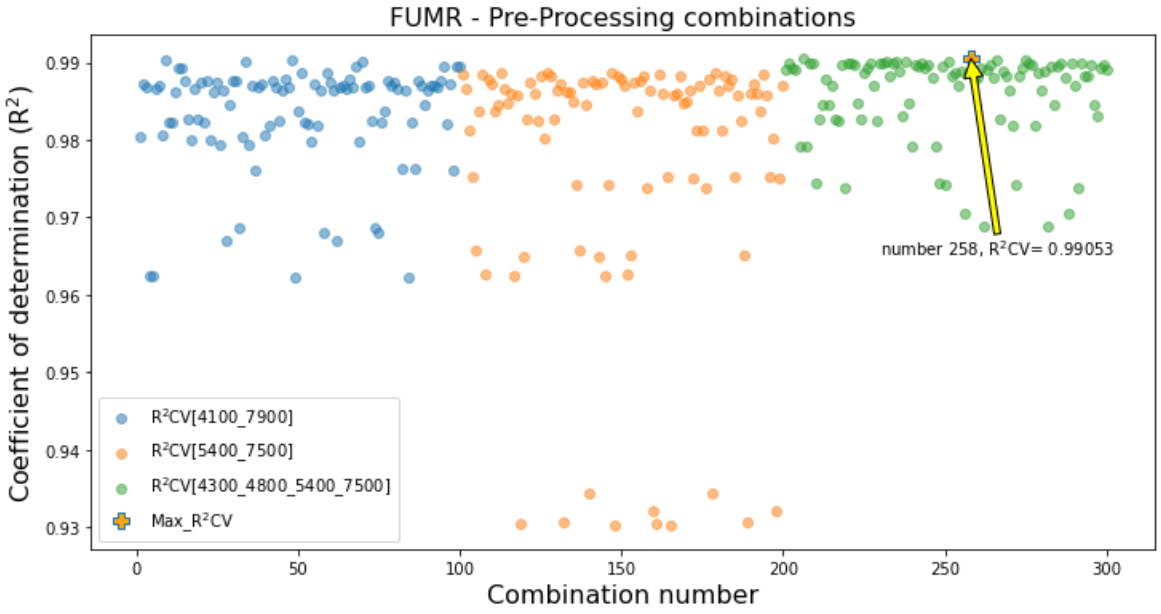


Figure 24 All PLS models generated to predict the FUMR in Python. Plot of the R^2CV vs the number of the specific preprocessing combination.

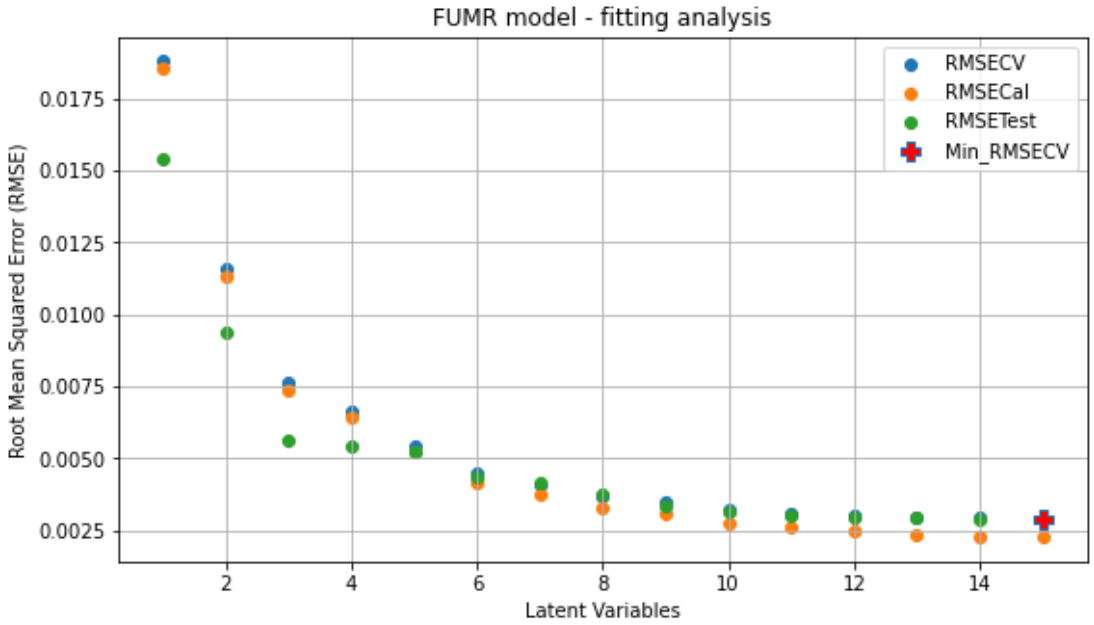


Figure 25 Graph of RMSE vs the number of latent variables. Fitting analysis of the best FUMR model and selection of the best number of LVs.

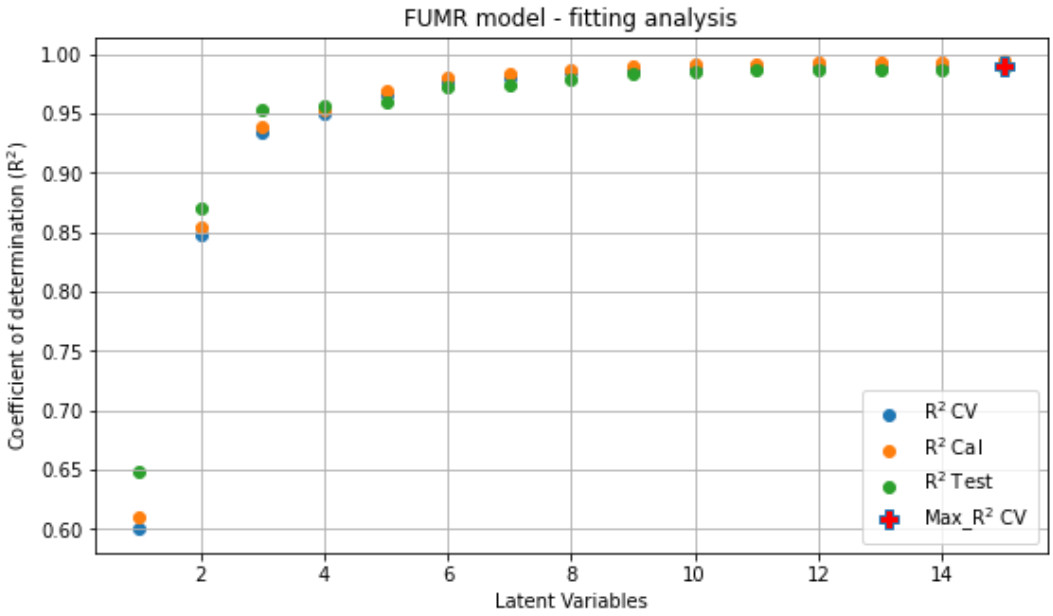


Figure 26 Graph of R^2 vs the number of latent variables. Fitting analysis of the best FUMR model and selection of the best number of LVs.

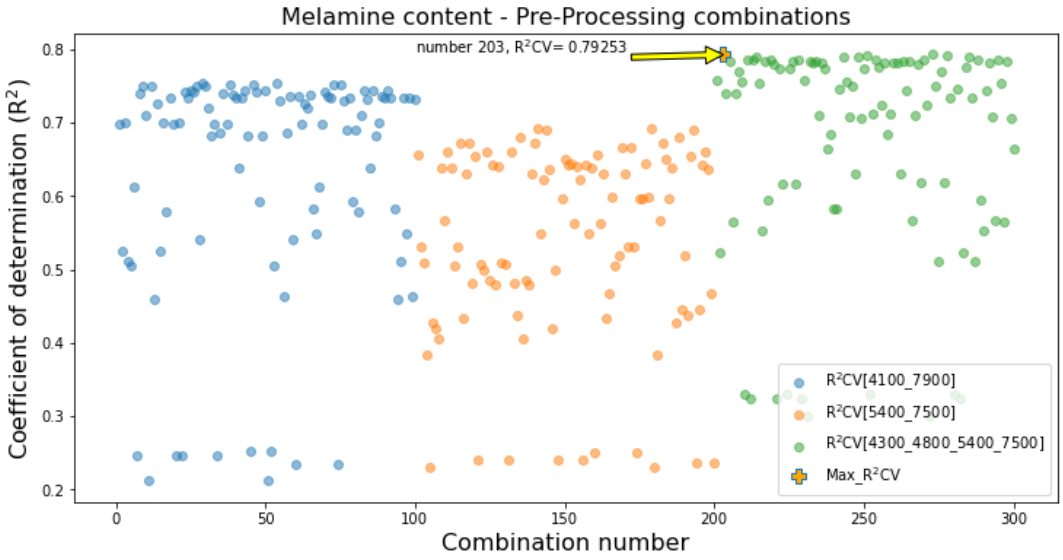


Figure 27 All PLS models generated to predict the melamine content in Python. Plot of the R^2CV vs the number of the specific preprocessing combination.

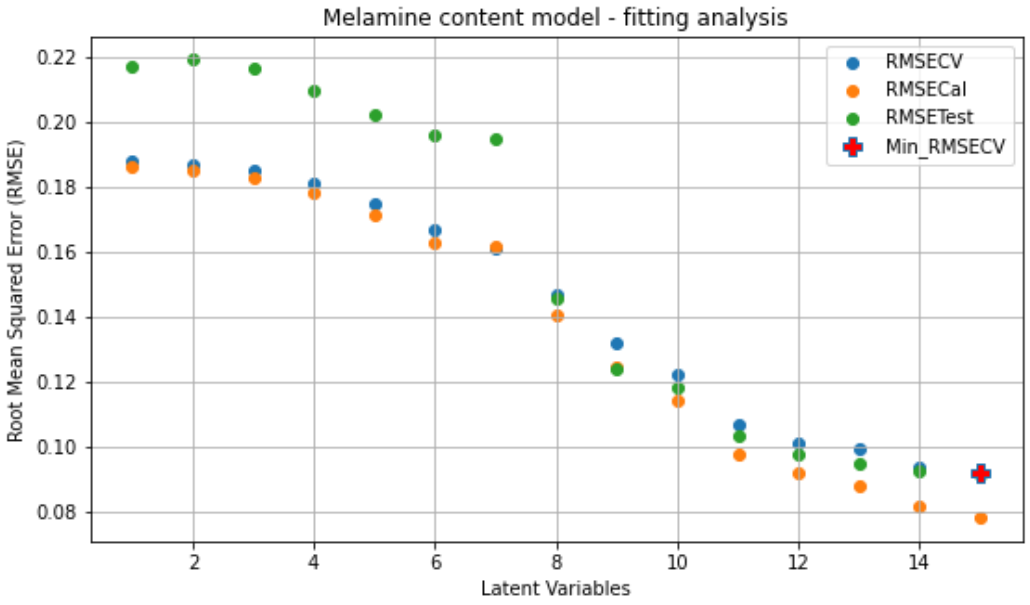


Figure 28 Graph of RMSE vs the number of latent variables. Fitting analysis of the best melamine content model and selection of the best number of LVs.

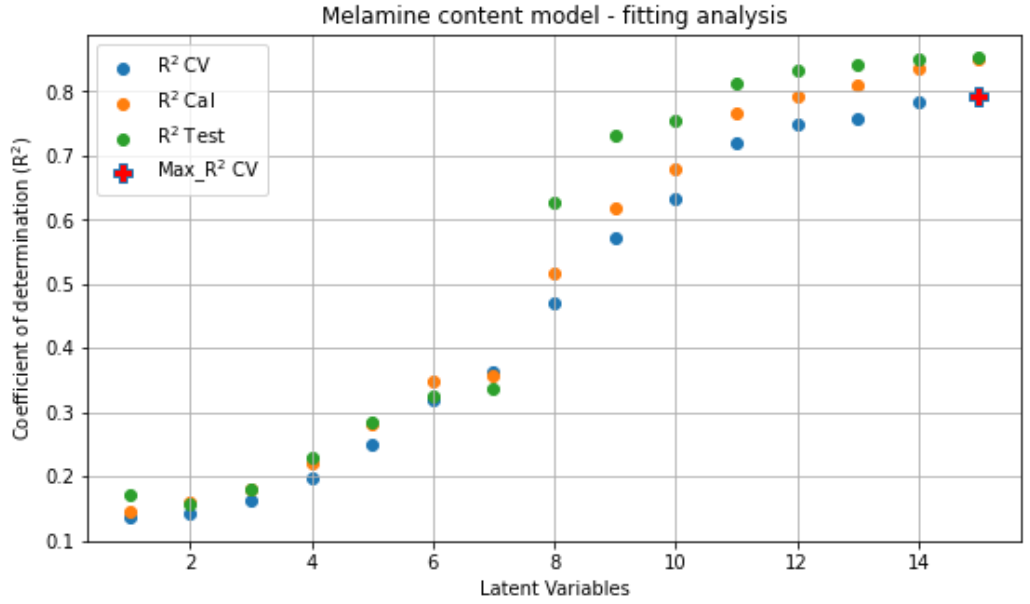


Figure 29 Graph of R^2 vs the number of latent variables. Fitting analysis of the best melamine content model and selection of the best number of LVs

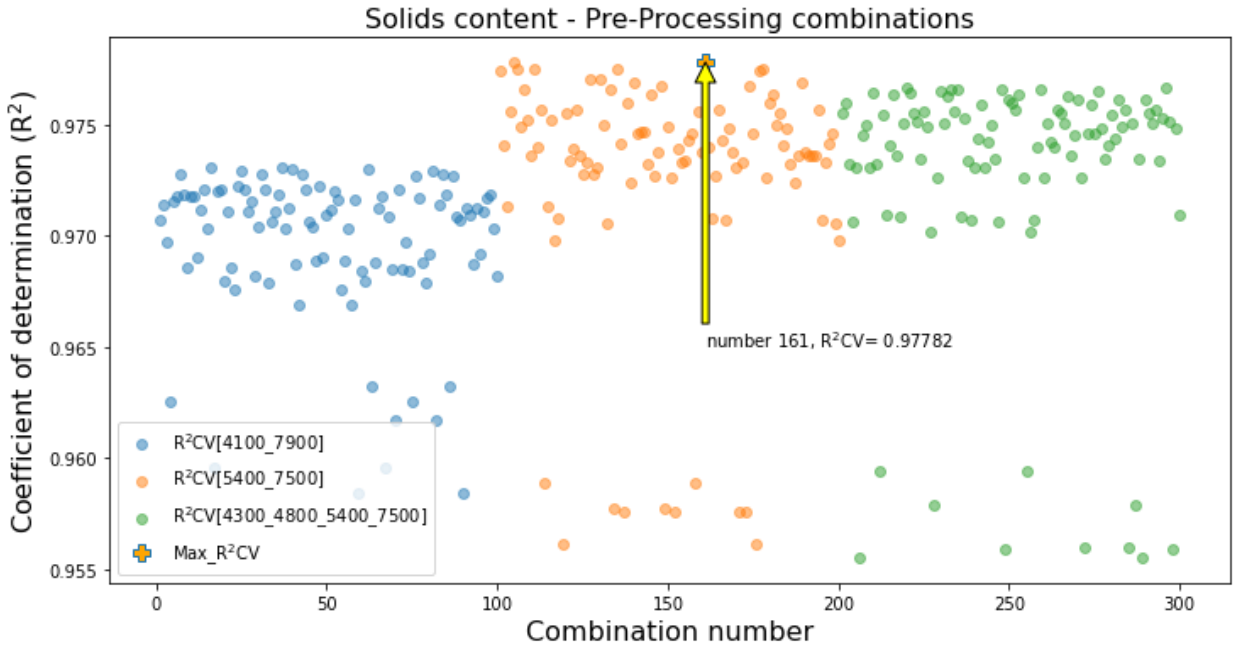


Figure 30 All PLS models generated to predict the SC in Python. Plot of the R^2CV vs the number of the specific preprocessing combination.

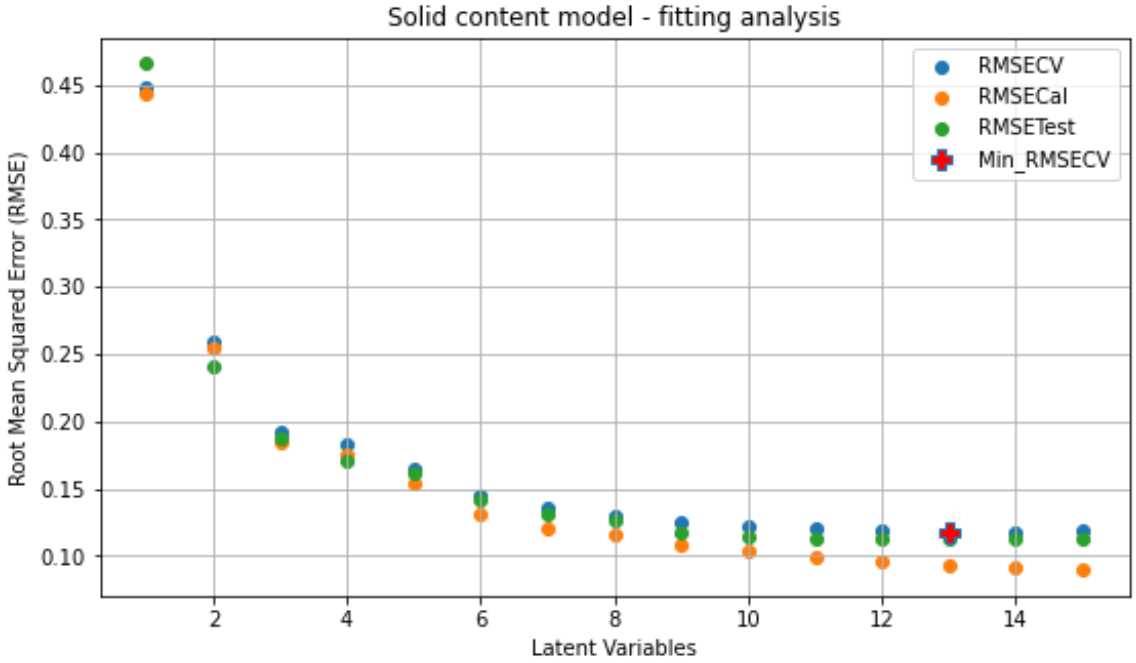


Figure 31 Graph of RMSE vs the number of latent variables. Fitting analysis of the best SC model and selection of the best number of LVs.

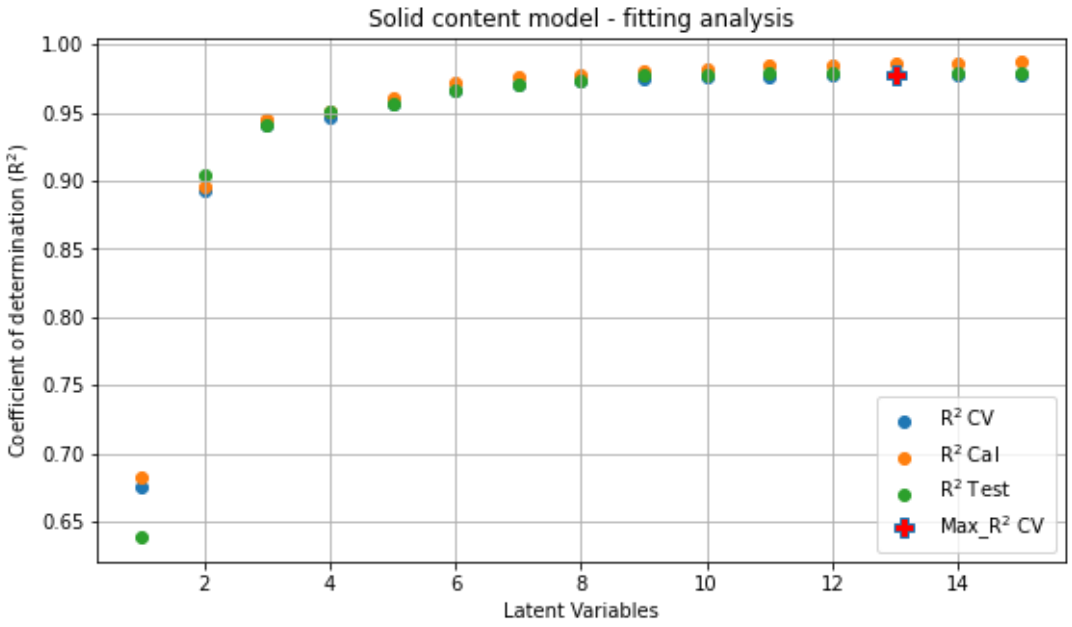


Figure 32 Graph of R^2 vs the number of latent variables. Fitting analysis of the best SC model and selection of the best number of LVs

Annex D - ANN code

```
# Loading Python packages

from sklearn.preprocessing import StandardScaler
from sys import stdout
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nippy
import linecache
import os
```

```
# Selection of the calibration set presented in excel files
#df_input= spectra + wavelengths, df_out= all properties

df_input=pd.read_excel('11G300 INPUT CAL.xlsx',header=None)
df_out=pd.read_excel('11G300 OUTPUT CAL.xlsx')

# Selection of the desired property

df_prop=df_out.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_cal=df_input.loc[0,:]
df_spect_cal=df_input.loc[1:,:]
```

```
# Selection of the test set presented in excel files
#df_input_test= spectra + wavelengths, df_out_test= all properties

df_input_test=pd.read_excel('11G300 INPUT TEST.xlsx',header=None)
df_out_test=pd.read_excel('11G300 OUTPUT TEST.xlsx')

# Selection of the desired property

df_prop_test=df_out_test.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_test=df_input_test.loc[0,:]
df_spect_test=df_input_test.loc[1:,:]
```

```
# NIPPY - Spectra Preprocessing to select the desired regions
    # 1. Load configuration presented in a '.ini' file extension

pipelines = nippy.read_configuration('MELAMINE_ANN_PIPELINE.ini')

    # 2. Load data - conversion to numpy

wavelength_cal = df_wave_cal.T.to_numpy() #Rows = wavelength, Columns = samples
wavelength_test = df_wave_test.T.to_numpy()
spectra_calibration = df_spect_cal.T.to_numpy()
spectra_test = df_spect_test.T.to_numpy()

    # 3. Dataset through all pipelines

datasets_calibration = nippy.nippy(wavelength_cal, spectra_calibration, pipelines)
datasets_test = nippy.nippy(wavelength_test, spectra_test, pipelines)

    # 4. Export the preprocessed data
nippy.export_pipelines_to_csv('PRE_PRO_MEL_ANN_CAL', datasets_calibration, \
pipelines, mkdir=True)
nippy.export_pipelines_to_csv('PRE_PRO_MEL_ANN_TEST', datasets_test, pipelines, mkdir=True)
```

```
# Function to automate Principal components analysis.
# cal_path - Location of the preprocessed spectra of the calibration set - csv file
# cal_pipelines - Location of the preprocessing type performed in calibration set - txt file
# pcs_number - maximum number of pcs to analyze

def pc_analysis(cal_path,cal_pipelines,pcs_number):
    # Number of pipelines created - equal to calibration and test sets

    with open(cal_pipelines, 'r') as fp:
        for count, line in enumerate(fp):
            pass
        print("Number of pipelines:",count+1)
        line_numbers = range(1,count+2)

        # Walk through the preprocessed calibration sets directories

        for (root1, dirs1, files1) in os.walk(cal_path):

            # Selection of the preprocessed data of the calibration sets

            for i,k in zip(sorted(files1,key=len),line_numbers):

                # Print the preprocessing type performed in the calibration set

                line1 = linecache.getline(cal_pipelines, k).strip()
                print('CAL_PIPELINE:',line1)
```

```
                # Calibration set after preprocessing

                data1 = np.genfromtxt(os.path.join(root1, i), delimiter=',')
                spec_cal = data1[:,1:]  # Rows = wavelength, Columns = samples
                wave1 = data1[:,0]
                print(i) # print file name

                # Normalization of x variables

                X_cal=spec_cal.T
                scx = StandardScaler()
                scx.fit(X_cal)
                X_calN=scx.transform(X_cal)

                # PCA

                xi=np.arange(1,pcs_number+1,1) # total number of pcs to analyze
                EV=[]
                for i in xi:
                    pca=PCA(n_components=i)
                    pca.fit(X_calN)
                    ev=pca.explained_variance_ratio_.sum()
                    EV.append(ev)

                # Data frame with the results

                df_pca=pd.DataFrame(EV,columns =['Expl. Variance'], index=xi).T
                display(df_pca)

                # PCA PLOT (EXPLAINED VARIANCE vs PCs)

                plt.figure(figsize=(10,5))
                plt.scatter(xi,EV)
                plt.title('PCA',size=14)
                plt.xlabel('Components', size=14)
                plt.ylabel('Explained Variance',size=14)
                plt.grid()
                plt.show()
```

```
# Example of how to apply the PCA function

cal_path='C:\\Users\\selen\\PRE_PRO_MEL_ANN_CAL'
cal_pipelines='C:\\Users\\selen\\PRE_PRO_MEL_ANN_CAL\\pipelines.log'
pcs=10
pc_analysis(cal_path,cal_pipelines,pcs)
```

```
# Function to automate ANN models' creation.
# cal_path - Location of the preprocessed spectra of the calibration set - csv files
# test_path - Location of the preprocessed spectra of the test set - csv files
# cal_pipelines - Location of the preprocessing type performed in calibration set - txt file
# test_pipelines - Location of the preprocessing type performed in test set - txt file
# neurons_number - number of neurons to analyze in the hidden layer
# pc_number - number of pcs to reduce the data

def ann_analysis(cal_path,test_path,cal_pipelines,test_pipelines,neurons_number,pc_number):
    # Number of pipelines created - equal to calibration and test sets

    with open(cal_pipelines, 'r') as fp:
        for count, line in enumerate(fp):
            pass
        print("Number of pipelines:",count+1)
        line_numbers = range(1,count+2)

        # Walk through the preprocessed calibration and test sets directories

        for (root1, dirs1, files1),(root2,dirs2,files2) in zip(
            os.walk(cal_path),os.walk(test_path)):

            # Selection of the preprocessed data of the calibration and test sets

            for i,j,k in zip(sorted(files1,key=len),sorted(files2,key=len),line_numbers):

                # Print the preprocessing type performed in the calibration set

                line1 = linecache.getline(cal_pipelines, k).strip()
                print('CAL_PIPELINE:',line1)

                # Calibration set after preprocessing

                data1 = np.genfromtxt(os.path.join(root1, i), delimiter=',')
                spec_cal = data1[:,1:]  # Rows = wavelength, Columns = samples
                wave1 = data1[:,0]
                print(i) # print file name

                # Print the preprocessing type performed in the test set

                line2 = linecache.getline(test_pipelines, k).strip()
                print('TEST_PIPELINE:',line2)

                # Test set after preprocessing

                data2 = np.genfromtxt(os.path.join(root2, j), delimiter=',')
                spec_test = data2[:,1:]  # Rows = wavelength, Columns = samples
                wave2 = data2[:,0]
                print(j) # print file name
```

```
# Definition of x and y variables

X_cal=spec_cal.T
X_test=spec_test.T
y_cal=df_prop
y_test=df_prop_test

# Normalization of the x variables

scx= StandardScaler()
scx.fit(spec_cal.T)
X_calN=scx.transform(X_cal)
X_testN=scx.transform(X_test)

# Normalization of the y variables

scy= StandardScaler()
scy.fit(y_cal)
y_calN=scy.transform(y_cal)
y_testN=scy.transform(y_test)

# Coefficient of determination

R2cal=[]
R2test=[]
R2CV=[]

# root mean squared error

RMSEcal=[]
RMSECV=[]
RMSEtest=[]

# Analysis of the ANN model
# Total number of neurons to analyze in the hidden layer

xj=np.arange(1,neurons_number+1,1)
for j in xj:

    # Dimension reduction by PCA

    pca=PCA(n_components=pc_number)
    pca.fit(X_calN)
    xpcacalN=pca.transform(X_calN)
    xpcatestN=pca.transform(X_testN)
```

```
# Transformation of y variables data to a 1-D array

y_calN=y_calN.ravel()
y_testN=y_testN.ravel()


#Aplication of MLP regressor

regANN= (
MLPRegressor(hidden_layer_sizes=(j,),
activation='logistic',
solver='lbfgs', max_iter=10000).fit(xpcacalN,y_calN)
)

# Leaving-one-out cross-validation prediction

loo=LeaveOneOut()
y_CV=cross_val_predict(regANN, xpcacalN, y_calN, cv=loo,n_jobs=-1)

# Prediction of the properties by the model

y_calP=regANN.predict(xpcacalN)
y_testP=regANN.predict(xpcatestN)

#Transformation of a 1-D array to a Data Frame
y_testP=pd.DataFrame(y_testP)
y_calP=pd.DataFrame(y_calP)
y_CV=pd.DataFrame(y_CV)

# Scale back the data to the original representation

y_CV=scy.inverse_transform(y_CV)
y_calP=scy.inverse_transform(y_calP)
y_testP=scy.inverse_transform(y_testP)

# Calculate the scores

r2cal=r2_score(y_cal,y_calP)
r2CV=r2_score(y_cal,y_CV)
r2test=r2_score(y_test,y_testP)
R2cal.append(r2cal)
R2CV.append(r2CV)
R2test.append(r2test)
```

```
            # Calculate the root mean squared errors

            rmsecal=mean_squared_error(y_cal,y_calP,squared=False)
            rmseCV=mean_squared_error(y_cal,y_CV,squared= False)
            rmsetest=mean_squared_error(y_test,y_testP,squared= False)
            RMSEcal.append(rmsecal)
            RMSECV.append(rmseCV)
            RMSEtest.append(rmsetest)

            # Update status on the same line

            comp = 100*(j+1)/(neurons_number+1)
            stdout.write("\r%d%% completed" % comp)
            stdout.flush()
        stdout.write("\n")

        #print results
        rmseminCV = np.argmin(RMSECV) #position of the minimum RMSECV
        print("Cross-Validation - Suggested number of nodes for the maximum \
        R2CV: ", xj[rmseminCV])
        print('RMSE CV calib: %5.6f' % RMSEcal[rmseminCV])
        print('RMSE CV: %5.6f' % RMSECV[rmseminCV])
        print('RMSE test: %5.6f' % RMSEtest[rmseminCV])
        r2maxCV = np.argmax(R2CV) #position of the maximum R2CV
        print("Cross-Validation - Suggested number of nodes for the minimum \
        RMSECV: ", xj[r2maxCV])
        print('R2 calib: %5.6f' % R2cal[r2maxCV])
        print('R2 CV: %5.6f' % R2CV[r2maxCV])
        print('R2 test: %5.6f' % R2test[rmseminCV])
```

```
# Example of how to apply the function

cal_path='C:\\Users\\selen\\PRE_PRO_MEL_ANN_CAL'
cal_pipelines='C:\\Users\\selen\\PRE_PRO_MEL_ANN_CAL\\pipelines.log'
test_path= 'C:\\Users\\selen\\PRE_PRO_MEL_ANN_TEST'
test_pipelines="C:\\Users\\selen\\PRE_PRO_MEL_ANN_TEST\\pipelines.log"
neuron_number=30
components=4
ann_analysis(cal_path,test_path,cal_pipelines,test_pipelines,neuron_number,components)
```

Annex E - Complementary results for the ANN models

This annex presents the analysis of the best ANN models. The graphs below highlight the best number of neurons with a red cross for the minimum RMSECV and maximum R2CV.

In this section, the CV is represented by the abbreviation 'looVal' related to leaving-one-out cross-validation.

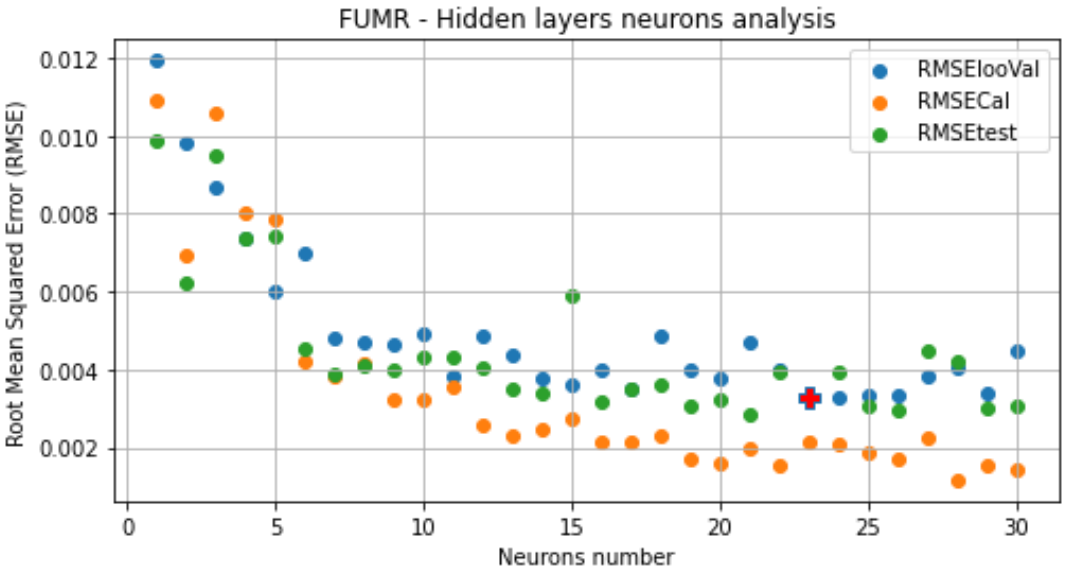


Figure 33 Graph of RMSE vs the number of neurons in the hidden layer. Fitting analysis of the best FUMR model and selection of the best neuron number.

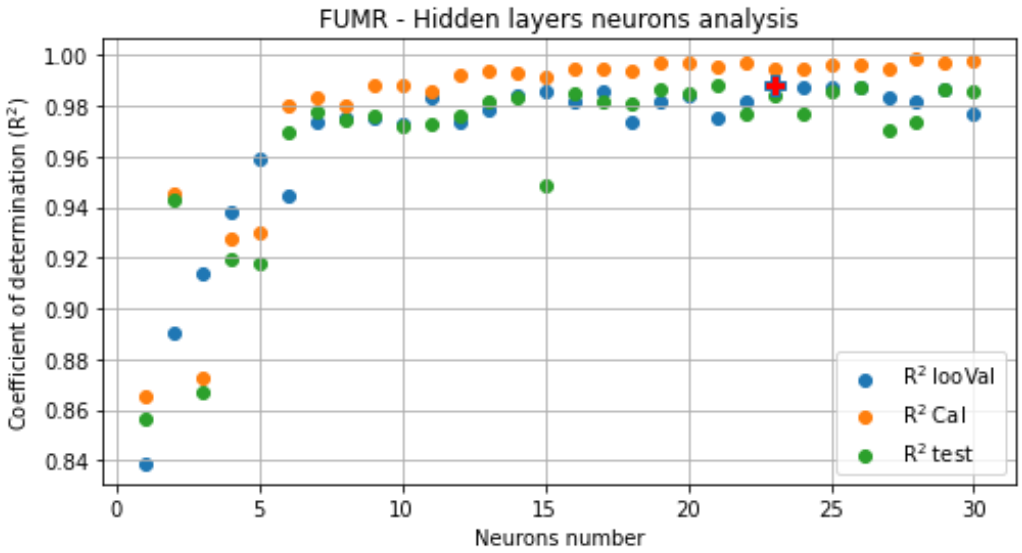


Figure 34 Graph of R^2 vs the number of neurons in the hidden layer. Fitting analysis of the best FUMR model and selection of the best neuron number

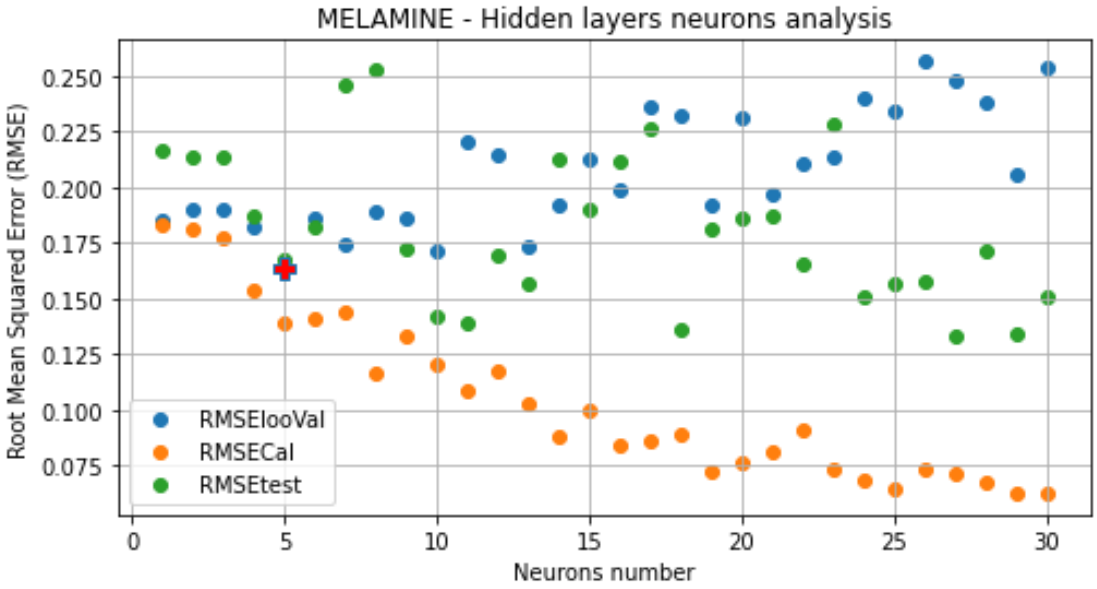


Figure 35 Graph of RMSE vs the number of neurons in the hidden layer. Fitting analysis of the best melamine content model and selection of the best neuron number.

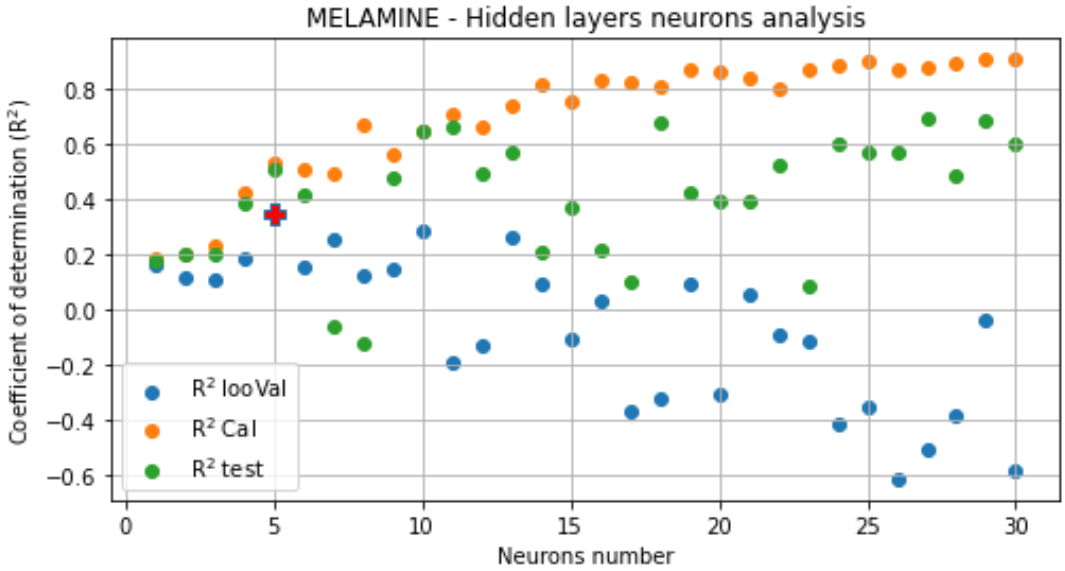


Figure 36 Graph of R^2 vs the number of neurons in the hidden layer. Fitting analysis of the best melamine content model and selection of the best neuron number

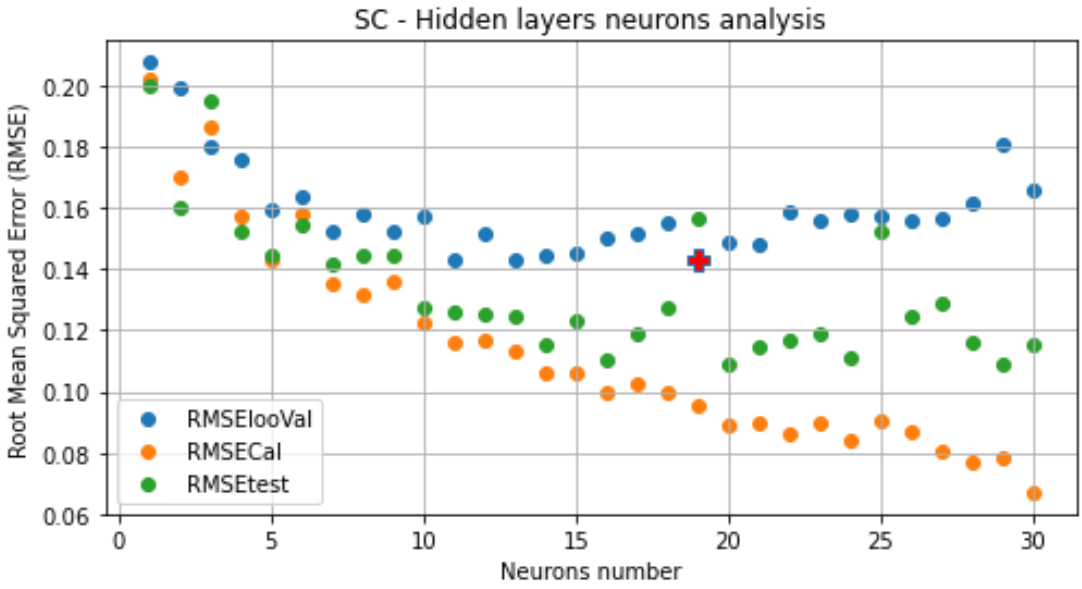


Figure 37 Graph of RMSE vs the number of neurons in the hidden layer. Fitting analysis of the best SC model and selection of the best neuron number.

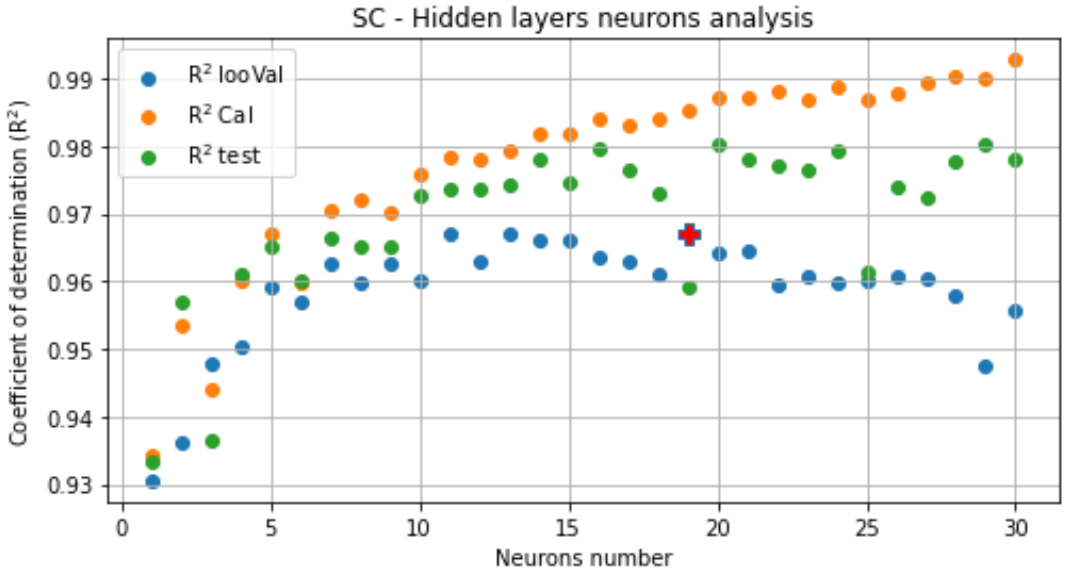


Figure 38 Graph of R^2 vs the number of neurons in the hidden layer. Fitting analysis of the best SC model and selection of the best neuron number.

Annex F - SVR code

```
# Loading Python packages

from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import LeaveOneOut
import nippy
import os
import linecache
import pandas as pd
import numpy as np
```

```
# Selection of the calibration set presented in excel files
# df_input= spectra + wavelengths, df_out= all properties

df_input=pd.read_excel('11G300 INPUT CAL.xlsx',header=None)
df_out=pd.read_excel('11G300 OUTPUT CAL.xlsx')

# Selection of the desired property

df_prop=df_out.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_cal=df_input.loc[0,:]
df_spect_cal=df_input.loc[1:,:]
```

```
# Selection of the test set presented in excel files
# df_input_test= spectra + wavelengths, df_out_test= all properties

df_input_test=pd.read_excel('11G300 INPUT TEST.xlsx',header=None)
df_out_test=pd.read_excel('11G300 OUTPUT TEST.xlsx')

# Selection of the desired property

df_prop_test=df_out_test.filter(['MELAMINE'])

# Selection of the wavelengths and spectra

df_wave_test=df_input_test.loc[0,:]
df_spect_test=df_input_test.loc[1:,:]
```

```
# NIPPY - Spectra Preprocessing to select the desired regions
    # 1. Load configuration presented in a '.ini' file extension

pipelines = nippy.read_configuration('MELAMINE_SVR_PIPELINE.ini')

    # 2. Load data - conversion to numpy

wavelength_cal = df_wave_cal.T.to_numpy() #Rows = wavelength, Columns = samples
wavelength_test = df_wave_test.T.to_numpy()
spectra_calibration = df_spect_cal.T.to_numpy()
spectra_test = df_spect_test.T.to_numpy()

    # 3. Dataset through all pipelines

datasets_calibration = nippy.nippy(wavelength_cal, spectra_calibration, pipelines)
datasets_test = nippy.nippy(wavelength_test, spectra_test, pipelines)

    # 4. Export the preprocessed data

nippy.export_pipelines_to_csv('PRE_PRO_MEL_SVR_CAL', datasets_calibration, pipelines,
mkdir=True)
nippy.export_pipelines_to_csv('PRE_PRO_MEL_SVR_TEST', datasets_test, pipelines, mkdir=True)
```

```
# Function to automate SVR models' creation.
# cal_path - Location of the preprocessed spectra of the calibration set - csv files
# test_path - Location of the preprocessed spectra of the test set - csv files
# cal_pipelines - Location of the preprocessing type performed in calibration set - txt file
# test_pipelines - Location of the preprocessing type performed in test set - txt file

def svr_analysis(cal_path,test_path,cal_pipelines,test_pipelines):
    # Number of pipelines created - equal to calibration and test sets

    with open(cal_pipelines, 'r') as fp:
        for count, line in enumerate(fp):
            pass
        print("Number of pipelines:",count+1)
        line_numbers = range(1,count+2)

        # Walk through the preprocessed calibration and test sets directories

        for (root1, dirs1, files1),(root2,dirs2,files2) in zip(
            zip(os.walk(cal_path),os.walk(test_path)):

            # Selection of the preprocessed data of the calibration and test sets

            for i,j,k in zip(sorted(files1,key=len), sorted (
                files2,key=len),line_numbers):
```

```
# Print the preprocessing type performed in the calibration set

line1 = linecache.getline(cal_pipelines, k).strip()
print('CAL_PIPELINE:',line1)

# Calibration set after preprocessing

data1 = np.genfromtxt(os.path.join(root1, i), delimiter=',')
spec_cal = data1[:,1:]  # Rows = wavelength, Columns = samples
wave1 = data1[:,0]
print(i) # print file name

# Print the preprocessing type performed in the test set

line2 = linecache.getline(test_pipelines, k).strip()
print('TEST_PIPELINE:',line2)

# Test set after preprocessing

data2 = np.genfromtxt(os.path.join(root2, j), delimiter=',')
spec_test = data2[:,1:]  # Rows = wavelength, Columns = samples
wave2 = data2[:,0]
print(j) # print file name

# Definition of x and y variables

X_cal=spec_cal.T
X_test=spec_test.T
y_cal=df_prop
y_test=df_prop_test

# Normalization of the x variables

scx= StandardScaler()
scx.fit(spec_cal.T)
X_calN=scx.transform(X_cal)
X_testN=scx.transform(X_test)

# Normalization of the y variables

scy= StandardScaler()
scy.fit(y_cal)
y_calN=scy.transform(y_cal)
y_testN=scy.transform(y_test)
```

```
# Coefficient of determination

R2cal=[]
R2test=[]
R2CV=[]

# root mean squared error

RMSEcal=[]
RMSECV=[]
RMSEtest=[]

# Transformation of y variables data to a 1-D array

y_calN=y_calN.ravel()
y_testN=y_testN.ravel()

#Aplication of the support vector machines regression

regsvr = SVR(kernel="rbf", C=1000, epsilon=0.001,gamma='scale')
regsvr.fit(X_calN, y_calN)

# Leaving-one-out cross-validation prediction

loo=LeaveOneOut()
y_CV=cross_val_predict(regsvr, X_calN, y_calN, cv=loo,n_jobs=-1)

#Prediction of the properties by the model

y_calP=regsvr.predict(X_calN)
y_testP=regsvr.predict(X_testN)

#Transformation of a 1-D array to a Data Frame

y_testP=pd.DataFrame(y_testP)
y_calP=pd.DataFrame(y_calP)
y_CV=pd.DataFrame(y_CV)

# Scale back the data to the original representation

y_CV=scy.inverse_transform(y_CV)
y_calP=scy.inverse_transform(y_calP)
y_testP=scy.inverse_transform(y_testP)
```

```
        # Calculate the scores

        r2cal=r2_score(y_cal,y_calP)
        r2CV=r2_score(y_cal,y_CV)
        r2test=r2_score(y_test,y_testP)
        R2cal.append(r2cal)
        R2CV.append(r2CV)
        R2test.append(r2test)

        # Calculate the root mean squared error

        rmsecal=mean_squared_error(y_cal,y_calP,squared=False)
        rmseCV=mean_squared_error(y_cal, y_CV,squared=False)
        rmsetest=mean_squared_error(y_test,y_testP,squared=False)
        RMSEcal.append(rmsecal)
        RMSEtest.append(rmsetest)
        RMSECV.append(rmseCV)

        # print results

        print('RMSE cal:', RMSEcal)
        print('RMSE CV:', RMSECV)
        print('RMSE test:', RMSEtest)
        print('R2 cal', R2cal)
        print('R2 CV:', R2CV)
        print('R2 test:', R2test)
```

```
# Example of how to apply the function

cal_path='C:\\Users\\selen\\PRE_PRO_MEL_SVR_CAL'
cal_pipelines='C:\\Users\\selen\\PRE_PRO_MEL_SVR_CAL\\pipelines.log'
test_path= 'C:\\Users\\selen\\PRE_PRO_MEL_SVR_TEST'
test_pipelines="C:\\Users\\selen\\PRE_PRO_MEL_SVR_TEST\\pipelines.log"
svr_analysis(cal_path,test_path,cal_pipelines,test_pipelines)
```